Tampereen yliopisto

Santeri Tuomisto

KUORMANTASAUSALGORITMI

Informaatioteknologian ja viestinnän tiedekunta
Pro gradu -tutkielma
Toukokuu 2019

TIIVISTELMÄ

Santeri Tuomisto: Kuormantasausalgoritmi
Pro gradu -tutkielma
Tampereen yliopisto
Tietojenkäsittelytieteiden tutkinto-ohjelma
Toukokuu 2019

Suomessa on asetettu kovia ilmastotavoitteita, joiden saavuttamiseksi nykyisen autokannan tulisi uusiutua ja korvautua pääasiassa sähköajoneuvoilla. Kuitenkaan harva nykyisistä omakotitaloista tai taloyhtiöiden parkkipaikoista on suunniteltu kestämään sähköautojen latausta. Ongelmia sulakkeiden kanssa tulee varsinkin silloin, jos parkkipaikalla on useampi sähköajoneuvo samaan aikaan latautumassa.

Tämä ongelma voidaan ratkaista käyttämällä tässä tutkielmassa toteutettua kuormantasausalgoritmia, joka tarvittaessa laskee tai nostaa vaiheen kokonaisvirrankulutusta säätelemällä sähköautojen latauslaitteita sekä henkilöautojen esilämmitystolppia. Kuormantasausalgoritmin tavoitteena on mahdollistaa autoille aina mahdollisimman suuri latausteho sulakkeen sallimissa rajoissa. Yksinkertainen kuormantasausalgoritmi jakaa latausvirran tasan latauksessa olevien autojen kesken. Tässä tutkielmassa toteutettu algoritmi rajoittaa latauslaitteiden virrankulutusta progressiivisesti sen mukaan kuinka paljon latauslaite kuluttaa virtaa.

Tutkielmassa esitellään kuormantasausongelmaa, laitteita, käytettyjä tekniikoita sekä logiikkaa, jolla algoritmi toimii. Tämän lisäksi algoritmin perustoimintoja testataan ja analysoidaan mahdollisia ongelmatilanteita ja kehityskohteita. Algoritmin toimintaperiaate on myös mahdollista viedä toisenlaiseenkin ympäristöön koskemaan muitakin kuin vain latauslaitteita ja esilämmitystolppia. Eräs käyttökohde voisi olla esimerkiksi omakotitalo, jossa sähköjärjestelmä ei kestä sähkökiuasta ja lattialämmitystä samaan aikaan. Tällöin algoritmi voisi rajoittaa lattialämmityksen virrankulutusta, kun sähkökiuas menee päälle.

Algoritmi on toteutettu Jidoka Technologies Oy:lle.

Avainsanat: sähköauto, kuormantasaus, kuormanhallinta, algoritmi, sähköautojen lataus.

Sisällys

Sanasto

BEV (Battery electric vehicle)
Täyssähköauto, eli auto, jonka voimanlähteenä on sähkömoottori, joka saa virtaa ladattavista akuista [Motiva 2015, s. 5-8].

EVSE/EVSE-tolppa (Electric vehicle supply equipment)
Sähköajoneuvojen latausasema.

HEAT/HEAT-tolppa
Esilämmitystolppaan tuleva pistorasia, jonka päällä oloa voidaan ohjata etänä. Eräänlainen älypistorasia, joka on suunniteltu nimenomaan autojen esilämmittämiseen. Sama laite kuin POW, mutta HEAT-tolpassa on 2 tunnin maksimipäälläoloaika.

MQTT (Message Queuing Telemetry Transport)
Kevyt viestintäprotokolla, joka toimii julkaisija-tilaaja mallilla [Hillar 2017, s. 9].

PHEV (Plug-in hybrid electric vehicle)
Lataushybridiauto, eli auto, jonka voimalähteenä on sähkömoottorin lisäksi myös polttomoottori. Erona hybridiautoon se, että ajoakkuun voidaan ladata sähköä myös verkkovirrasta. [Motiva 2015, s. 5-8.]

POW/POW-tolppa
Esilämmitystolppaan tuleva pistorasia, jonka päällä oloa voidaan ohjata etänä. Eräänlainen älypistorasia, joka on suunniteltu nimenomaan autojen esilämmittämiseen tai sähköautojen hitaaseen lataamiseen. Sama laite kuin HEAT, mutta POW-tolpassa ei ole maksimipäälläoloaikaa.

1 Johdanto

Ilmastonmuutos ja liikenne ovat tällä hetkellä ihmisten huulilla. Varsinkin sähköä pidetään tulevaisuuden henkilöautoliikenteen pääasiallisena voimanlähteenä. Tähän vaikuttavat esimerkiksi Euroopan Unionin asettamat ilmastotavoitteet, joiden seurauksena moni maa on asettanut tavoitteita sähköautojen määrissä. Varsinkaan vanhojen parkkipaikkojen tai omakotitalojen sähköratkaisuja ei ole suunniteltu kestämään sähköautojen lataamista. Sähköautoistumisesta voi seurata taloyhtiöille kalliita remontteja, mikäli parkkipaikat täytyy rakentaa uudelleen. Remontille on olemassa kuitenkin halvempi vaihtoehto: algoritmi, joka säätelee parkkipaikan kuormaa, jolla vältytään sähköverkon ylikuormittumiselta ja sulakkeen palamiselta. Tällaista algoritmia kutsutaan kuormantasausalgoritmiksi.

Kuormantasausalgoritmin toteutus valikoitui tutkielman aiheeksi siten, että sain aiheen työpaikaltani. Algoritmi on siis toteutettu Jidoka Technologies Oy:lle. Tätä kautta tutkielman algoritmille tuli tehtävänanto, joka määritti, miten algoritmin tulisi toimia. Tehtävänanto ei kuitenkaan ollut täysin joustamaton, jolloin pystyin myös tekemään omia ratkaisujani algoritmin toteutuksessa. Työn tavoitteena oli toteuttaa algoritmi, joka täyttää sille asetetut tavoitteet. Pääpiirteittäin algoritmin tulisi tarvittaessa osata laskea ja nostaa vaiheen virrankulutusta säätelemällä vaiheen latauslaitteita ja esilämmitystolppia. Aihe on mielenkiintoinen ajankohtaisuuden vuoksi, sillä tällä hetkellä monessa taloyhtiössä pohditaan, miten latausratkaisut toteutetaan sähköautojen yleistyessä.

Vaikka kuormantasaus, sähköautot ja niiden lataus ovat erittäin ajankohtaisia aiheita, on silti vaikea löytää täysin vastaavanlaisia toteutuksia sisältäviä tieteellisiä artikkeleita tai tutkimuksia. Eräässä tutkimuksessa kuitenkin toteutettiin algoritmi, jonka tavoitteena on lieventää latauskuormien aiheuttamaa vaikutusta sähköverkkoon mahdollistamalla latauslaitteiden hallinnan reaaliajassa ja etänä. Algoritmi ottaa huomioon sekä jakeluverkon omistajien että latauslaitteiden käyttäjien mieltymykset. Tutkimus osoitti, että tehty algoritmi suoriutui hyvin ja pystyi hallitsemaan latauslaitteita halutulla tavalla. [Zuccaro *et al.* 2014.]

Sähköautojen latauksen optimointi on tutkittu aihe, mutta hieman eri näkökulmasta. Eräässä tutkimuksessa toteutettiin algoritmi, jonka avulla pyrittiin erilaisiin tavoitteisiin optimoimalla sähköautojen lataussyklejä. Tavoitteet olivat taloudelliset säästöt, suurempi tuulivoiman käyttö, pienempi saastuttavien energiamuotojen käyttö sekä kuorman pienentäminen virrankulutuspiikkien aikana. Kaikki optimointitavoitteet johtivat positiivisiin tuloksiin: esimerkiksi hintoihin perustuva optimointi tuotti noin 8-9 euron lataussäästöt 60 euron latauksessa. [Finn *et al.* 2012.]

Optimointiin liittyen on myös esimerkiksi tutkittu, miten sähköautojen lataamisen hintaa voitaisiin optimoida koneoppimisen avulla ilman, että tarvitsee tehdä ennustuksia

muuttujista, kuten sähköhinnasta tai latauksen kestosta. Lataamisen optimointi koneoppimisen avulla tuotti noin 88% rahallisen säästön verrattuna polttomoottoriauton kuluihin. Myös huomattiin, että mitä monimutkaisempi koneoppimisen tekniikka oli käytössä, sitä suuremmat säästöt saatiin. [Lopez *et al*. 2018.]

On myös tutkittu ja toteutettu algoritmeja kodin sähköjärjestelmien eli kodinkoneiden energianhallintaan, jotta saadaan vähennettyä virrankulutuspiikkejä ja lisättyä kodin energiatehokkuutta [Elma *et al.* 2015]. Tämän lisäksi on tehty tutkimuksia mikrosähköverkoista, joissa tuotetaan uusiutuvaa energiaa energianlähteenä ja käytetään sähköautoja energiavarastoina. Tutkimuksessa huomattiin, että tällaisen mikrosähköverkon riippuvuus sähköverkosta on pienempi ja että sähköautot pystyttiin lataamaan 100% puhtaalla energialla. [Mesaric *et al.* 2015.]

Sähköautot tarjoavatkin useita mahdollisuuksia. Niitä voidaan käytetään energiavarastoina, joista energiaa voidaan ottaa takaisin sähköverkkoon piikkien aikana. On myös mahdollista ladata sähköauton akut yösähköllä, joka on halvempaa ylituotannon takia ja käyttää tätä halvempaa sähköä päivällä esimerkiksi kodin sähköntarpeen täyttämiseen. Aurinkopaneelit ja sähköautot ovat myös yhteensopiva kombinaatio. Tässä työssä kuitenkin tutkitaan asiaa vain sähköautojen virrankulutuksen ja sen säätelyn näkökulmasta. Tavoitteena on toteuttaa algoritmi, joka hallitsee samassa vaiheessa olevien sähköautojen latausvirtoja niin, ettei sulakkeet pala. Ongelma ei kuulosta erikoiselta, mutta kuten aikaisemmin tässä luvussa todettiin, monet taloyhtiöt ja omakotitalojen omistajat painivat tämän ongelman kanssa, sillä harva nykyisistä sähköratkaisuista kestää yhden tai varsinkaan useamman sähköauton lataamisen.

Aluksi, luvussa 2, esitellään ladattavia sähköajoneuvoja ja näiden erilaisia lataustapoja. Luvussa tehdään katsaus, paljonko Suomessa tällä hetkellä on sähköajoneuvoja sekä käsitellään Suomen asettamia sähköautojen tavoitemääriä. Tämän lisäksi luvussa esitellään laitteita, joita työssä käytetään. Ensimmäinen laitteista on pistorasia, joka tulee esilämmitystolppaan ja toinen on sähköautojen lataukseen tarkoitettu latauslaite. Luvussa 3 esitellään ongelmaa, jonka tämän työn algoritmi pyrkii ratkaisemaan, sekä algoritmin oikeellista toimintaa. Luvussa on myös kaksi parkkipaikkaskenaariota, joissa esitellään algoritmin toimintaa. Luvussa 4 kerrotaan algoritmin toteutuksesta, muun muassa käymällä läpi käytettyjä tekniikoita ja esittelemällä algoritmin logiikkaa vuokaavioiden avulla. Luvussa 5 on algoritmin testausta. Luvussa algoritmille asetetaan vaatimuksia ja määritellään testejä, minkä jälkeen testit suoritetaan ja tulokset raportoidaan. Luvussa 6 analysoidaan algoritmin mahdollisia ongelmatilanteita ja esitellään parannusehdotuksia. Tämän lisäksi luvussa pohditaan ratkaisun skaalautuvuutta sekä mahdollisia muita algoritmin käyttökohteita. Lopuksi luvussa luodaan katsaus muihin tällä hetkellä markkinoilla oleviin kuormantasausratkaisuihin. Luvussa 7 on yhteenveto koko työstä.

2 Ladattavat sähköajoneuvot ja latauslaitteet

Ladattavalla sähköajoneuvolla tarkoitetaan kulkuvälinettä, jossa on verkkovirralla ladattava ajoakku, jonka sähköenergiaa käytetään ajoneuvoa liikuttavaan sähkömoottoriin. Nämä ajoneuvot voidaan jakaa seuraaviin ryhmiin [Motiva 2015, s. 8.]:

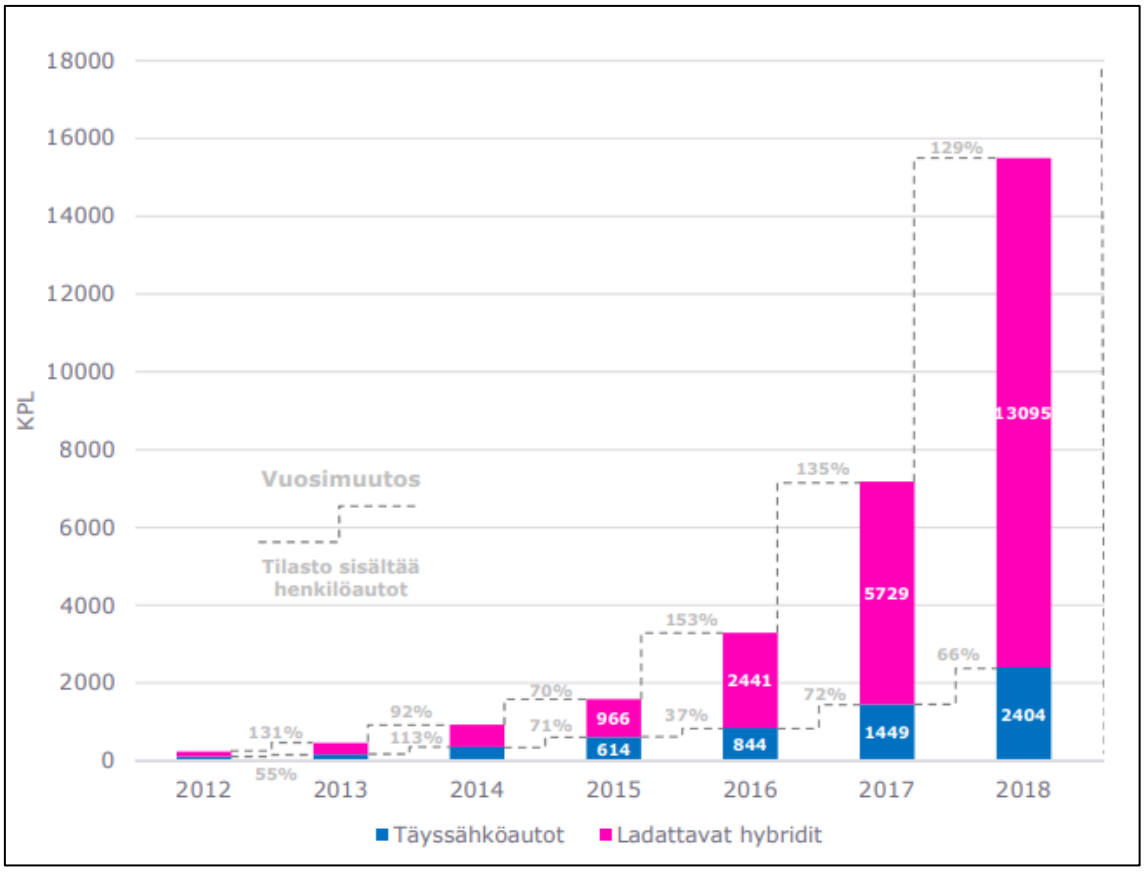
1. Kevyet sähkökäyttöiset ajoneuvot, kuten esimerkiksi sähköpyörät tai sähkömopot
2. Lataushybridiautot (PHEV/Plug-in hybrid electric vehicle) eli ajoneuvot, joiden voimalähteenä on sähkömoottorin lisäksi myös polttomoottori. Erona hybridiautoon se, että ajoakkuun voidaan ladata sähköä myös verkkovirrasta
3. Täyssähköautot (BEV/Battery electric vehicle)

Seuraavaksi käsitellään EU:n asettamia päästötavoitteita, sähköautojen määrää Suomessa ja muualla maailmassa, sekä sähköautojen ja latauslaitteiden tavoitemääriä, joiden tulisi täyttyä, jotta ilmastotavoitteisiin päästäisiin. Seuraava kohta valottaa myös sitä, kuinka paljon tulevaisuudessa kuormantasausalgoritmien kysyntä mahdollisesti tulee kasvamaan latauslaitteiden määrän kasvun seurauksena.

2.1 Sähköautojen määrä Suomessa ja muualla maailmassa

Lokakuussa 2014 Eurooppa-neuvostossa eli EU:n huippukokouksessa sovittiin uusista päästövähennystavoitteista. Tavoitteena on kasvihuonepäästöjen vähentäminen 40 prosentilla EU:ssa vuoteen 2030 mennessä verrattuna vuoteen 1990. Kaikista Suomen kasvihuonepäästöistä liikenne tuottaa noin viidenneksen, joka tarkoitti vuonna 2015 noin 11 miljoonaa tonnia hiilidioksidiekvivalenttia. Suomessa liikenteen päästöistä noin 90 prosenttia syntyy tieliikenteestä. Tieliikenteen päästöistä 58 prosenttia syntyy henkilöautoliikenteestä, 37 prosenttia paketti- ja kuorma-autoista ja loput muista ajoneuvoista, kuten esimerkiksi busseista ja moottoripyöristä. Suomen tavoitteena on olla tieliikenteen osalta lähes nollapäästöinen vuonna 2050. Tämän saavuttamiseksi sähkökäyttöisten henkilöautojen tavoitemäärä vuonna 2020 on 20 000 autoa ja vuonna 2030 on 250 000 autoa. EU:n vuonna 2014 asettaman jakeluinfradirektiivin suositus on 1 julkinen latauslaite 10 sähköautoa kohden, joka tarkoittaa 25 000 latauslaitetta vuonna 2030, mikäli 250 000 sähköauton tavoitemäärä täyttyy. [Valtioneuvosto 2017, ss. 3-24.]

Vuoden 2018 lopussa Suomessa oli 15 499 kappaletta sähköautoja, joista 2 404 oli täyssähköautoja ja 13 095 oli lataushybridiautoja. Vuoden 2018 aikana sähköautoja rekisteröitiin 8 321 kappaletta, joista 955 oli täyssähköautoja ja 7 336 oli lataushybridiautoja. Vuoden 2020 lopun välitavoite 20 000 sähköautolle tulee ylittymään todennäköisesti jo alkuvuodesta 2019. Suomen sähköhenkilöautokannan kehitys on kokonaisuudessaan nähtävissä kuvassa 1. [Teknologiateollisuus 2019, ss. 3-7.]



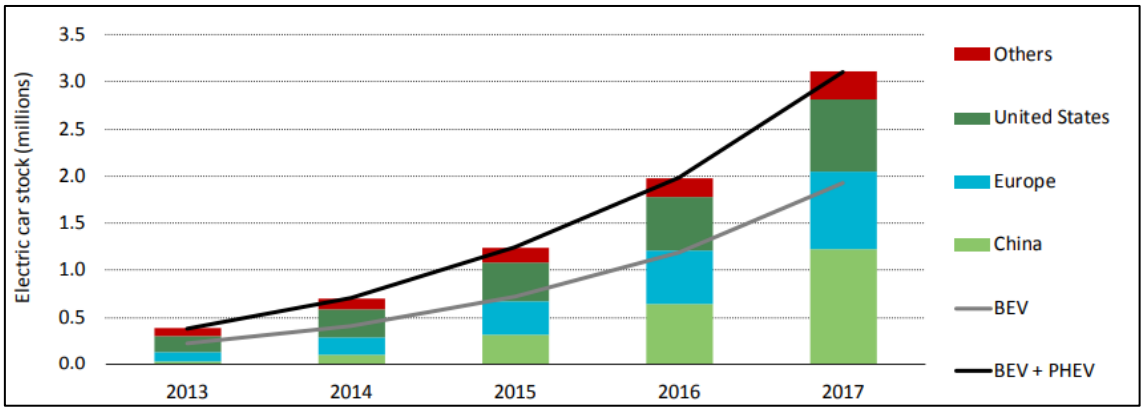
Kuva 1. Suomen sähköhenkilöautokannan kehitys [Teknologiateollisuus 2019, s. 15].

Maailmanlaajuisesti sähköautojen määrä ylitti arvioiden mukaan 3 miljoonaa autoa vuonna 2017 (kuva 2). Vuoden 2016 lopussa sähköautoja oli maailmanlaajuisesti 2 miljoonaa kappaletta, eli vuoden 2017 aikana sähköautojen määrä lisääntyi yli miljoonalla kappaleella. Tämä tarkoittaa noin 57 prosentin kasvua edeltävästä vuodesta. Vuoden 2015 ja 2016 välillä sähköautojen määrä kasvoi myöskin vastaavanlaisesti 60 prosenttia. Erään arvion mukaan vuonna 2030 sähköautoja olisi noin 120 miljoonaa kappaletta maailmanlaajuisesti. [IEA 2018, ss. 19-76.] Suomessa sähköautokannan kasvu on ollut maailmanlaajuista kasvua nopeampaa, sillä vuosien 2015 ja 2016 välillä Suomen sähköautokanta kasvoi 153 prosenttia ja vuosien 2016 ja 2017 välillä 135 prosenttia (kuva 1) [Teknologiateollisuus 2019, s. 15].

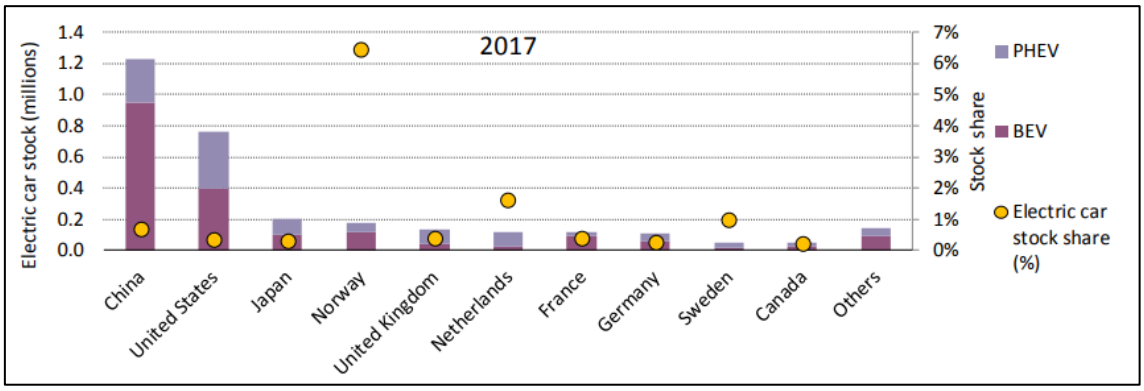
Vuonna 2018 Suomessa ensirekisteröitiin 212 täyssähköautoa ja 2535 lataushybridiautoa [Teknologiateollisuus 2019, ss. 18-19]. Suomessa ensirekisteröitiin vuonna 2018 yhteensä 120 499 henkilöautoa [Autoalan Tiedotuskeskus 2018]. Tämä tarkoittaa, että Suomessa vuonna 2018 ensirekisteröidyistä autoista noin 2.3 prosenttia oli sähköhenkilöautoja. Suomen liikennekäytössä oleva autokanta oli vuoden 2018 lopussa 2 696 334 autoa [Autoalan Tiedotuskeskus 2019]. Tämä tarkoittaa, että sähköautojen osuus Suomen koko autokannasta oli noin 0.6% vuoden 2018 lopussa.

Koko maailman sähköautokannasta noin 40 prosenttia, eli 1.2 miljoonaa sähköautoa, on Kiinassa (kuva 3), joka vastaa kuitenkin alle prosenttia koko Kiinan autokannasta. Yli

yhden prosentin sähköauto-osuuteen maan koko autokannasta pääsee vain kolme maata: Norja (6.4%), Hollanti (1.6%) ja Ruotsi (1.0%). [IEA 2018, s. 19.]



Kuva 2. Maailman sähköautokannan kehitys vuosittain [IEA 2018, s. 19].



Kuva 3. Sähköautojen määrät ja prosenttiosuudet koko autokannasta maittain [IEA 2018, s. 19].

2.2 Lataustavat

Ladattavien sähköajoneuvojen johdolliset lataustavat voidaan jakaa neljään luokkaan [Sesko 2018, ss. 1-2.]:

1. Lataustapa 1 / Kevyiden sähköajoneuvojen lataus
2. Lataustapa 2 / Hidas lataus
3. Lataustapa 3 / Peruslataus
4. Lataustapa 4 / Teholataus

Seuraavaksi käsitellään näitä lataustapoja tarkemmin. Esimerkkinä lataustapojen nopeudesta käytetään 25 kWh akkua. Tämä latausmäärä vastaa noin 125 km – 165 km toimintamatkaa sähköautolla tai lataushybridiautolla, sillä autot kuluttavat verkosta sähköä noin 15 – 20 kWh sataa ajokilometriä kohden. [Motiva 2015, s. 10.]

2.2.1 Lataustapa 1 / Kevyiden sähköajoneuvojen lataus

Lataustapa 1 on tarkoitettu pienitehoisten kevyiden sähkökäyttöisten ajoneuvojen, kuten sähkömopojen tai sähköpolkupyörien, lataamiseen. Lataustavassa käytetään tavallista kotitalouspistorasiaa. [Sesko 2018, s. 2.]

2.2.2 Lataustapa 2 / Hidas lataus

Lataustapa 2, eli hidas lataus, on sähköautojen lataustapa, jossa sähköautoa ladataan normaalista kotitalouspistorasiasta tai teollisuuspistorasiasta. Kotitalouspistorasia on suunniteltu kestämään 16 ampeerin virtaa vain kahden tunnin ajan, joten sähköajoneuvojen latausvirta tulee rajoittaa 8 ampeeriin kotitalouspistorasiaa käytettäessä. [Sesko 2018, s. 1.]

Tällainen lataus voi esimerkiksi tapahtua pihan lämmitystolpasta. Latausaika 25 kWh akulla tyhjästä täyteen 6 – 8 ampeerin latausvirralla vie noin 11 – 18 tuntia. [Motiva 2015, ss. 7-10.]

2.2.3 Lataustapa 3 / Peruslataus

Lataustapa 3, eli peruslataus, on sähköautojen suositeltavin lataustapa. Lataustavassa käytetään erityistä tyypin 2 sähköautopistorasiaa. Latausjärjestelmässä on mahdollisuus ohjata kuormitusta ja virran syöttöä. [Sesko 2018, s. 1.]

Peruslatausaseman latausvirta voi olla 0 – 16 ampeeria yhdellä vaiheella tai 0 – 32 ampeeria kolmella vaiheella. Esimerkiksi 16 ampeerin latausvirralla 25 kWh akun lataaminen tyhjästä täyteen kestää noin 7 tuntia. Peruslatauksessa latauskaapelina on joko tyypin 1 tai tyypin 2 latauskaapeli (kuva 4). [Motiva 2015, ss. 9-10.]

Kuva 4. Tyypin 1 ja tyypin 2 latauskaapelit [Plugit 2019].

2.2.4 Lataustapa 4 / Teholataus

Lataustapa 4, eli teho- tai pikalataus, on sähköajoneuvon akuston lataamista tasasähköllä suurella virralla [Sesko 2018, s. 2]. Pikalatauksella 25 kWh akun lataaminen tyhjästä täyteen vie noin 20 – 50 minuuttia [Motiva 2015, ss. 9-10].

Pikalatausasemien latausnopeudet tulevat kasvamaan tulevaisuudessa entisestään. Tällä hetkellä latausnopeudet ovat pikalatausasemilla 22 - 50 kW, mutta esimerkiksi marraskuussa 2018 Suomeen asennettiin ensimmäinen suurteholatausasema, jossa latausnopeus on 150 kW [Teknologiateollisuus 2018, s. 21]. Teholatauksessa latauskaapelina on joko CHAdeMO- tai 62196-3 Combo -latauskaapeli (kuva 5) [Motiva 2015, s. 10].

Kuva 5. Pikalatauskaapelit CHAdeMO ja 62196-3 Combo [Plugit 2019].

2.3 Laitteiden esittely

Seuraavaksi esitellään laitteet, joiden ohjaukseen algoritmi on alun perin suunniteltu.

2.3.1 Lämmitystolppa

Ensimmäinen näistä laitteista on autojen esilämmitystolppaan tuleva pistorasia, jonka tarkoituksena on korvata yleensä esilämmitystolpissa oleva ajastin (kuva 6). Pistorasiaan voidaan laittaa normaali töpseli, kuten esimerkiksi auton esilämmitysjohto. Pistorasia on jatkuvasti päällä ja yhteydessä verkkoon WiFi-yhteydellä. Pistorasiassa on virtanäppäin, jonka avulla voidaan säädellä, antaako se virtaa siinä kiinni oleville sähkölaitteille. Pistorasian virransyötön määrää ei ole mahdollista säädellä.

Pistorasiaa on mahdollista kontrolloida etänä esimerkiksi nettisivulta tai mobiiliapplikaatiosta. Näistä palveluista on virransyötön lisäksi mahdollista asettaa erilaisia ajastimia, joilla voidaan säätää milloin pistorasia käynnistyy. Pistorasian maksimipäälläoloaika on myös mahdollista asettaa esimerkiksi lämmitystolpille yleiseen 2 tuntiin.

Tämäntyyppistä pistorasiaa, joka on tarkoitettu vain auton esilämmittämiseen, kutsutaan jatkossa tässä työssä nimellä HEAT-tolppa tai vain nimellä HEAT. Pistorasiasta on kuitenkin myös mahdollista ladata sähköautoa. Lataustyyli pistorasiasta on lataustapa 2, eli hidaslataus, ja niitä pistorasioita, jotka tämän mahdollistavat, kutsutaan jatkossa nimellä POW-tolppa tai vain nimellä POW. Nämä molemmat nimitykset viittaavat yksittäiseen pistorasiaan. Niiden suurimpana erona on se, että HEAT-tolpalla on 2 tunnin päälläoloaikarajoitus, kun taas vastaavasti POW-tolpalla tätä aikarajoitusta ei ole.

Tämäntyyppinen etäohjattava pistorasia on myös mahdollista laittaa minkä tahansa sähkölaitteen ja pistorasian väliin, jolloin esimerkiksi olisi mahdollista saada etäohjattava kahvinkeitin.

Kuva 6. Esilämmitystolppaan tuleva pistorasia, jota voidaan ohjata etänä.

Pistorasian sisällä on sensoreita, jotka laskevat virrankulutusta. Tämä virrankulutustieto lähetetään algoritmille ja sen pohjalta algoritmi toimii. Pistorasian ja algoritmin viestiliikenteeseen mennään tarkemmin kohdassa 4.2.

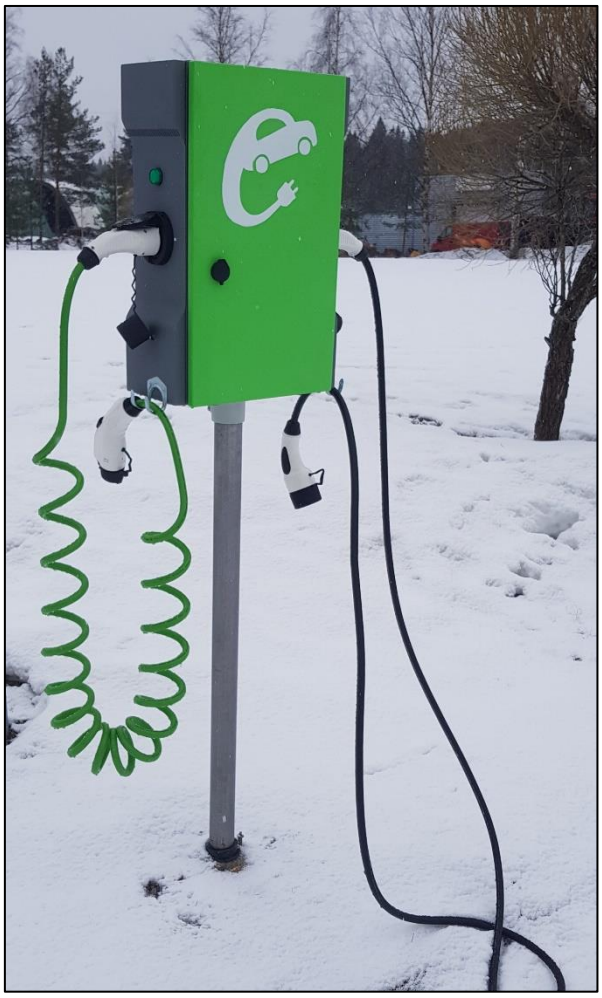
2.3.2 Latauslaite

Toinen laitteista on sähköautojen lataamiseen tarkoitettu latauslaite (kuva 7), jota jatkossa kutsutaan myös nimellä EVSE-tolppa tai vain EVSE (Electric vehicle supply equipment). Laitteessa käytetään tyypin 2 latauskaapeleita ja lataustapa on nopeudeltaan lataustapa 3 eli peruslataus. Latauslaite on yhteydessä verkkoon WiFi-yhteyden kautta, jonka kautta se kuuntelee komentoja ja lähettää viestejä. Laitteen lähettämään viestiliikenteeseen ja komentoihin palataan tarkemmin kohdassa 4.2. Latauslaitteiden kontrollointi on mahdollista samalta nettisivulta ja mobiiliapplikaatiosta, joista myös HEAT- ja POW-tolppia ohjataan. Näin käyttäjä voi laittaa latauslaitteen päälle ja pois etänä. Tämän lisäksi latauslaitteen on mahdollista säädellä virrankulutusta, jonka voi minimissään asettaa kuuteen ampeeriin. Latauslaitteen sisällä olevat sensorit laskevat ja seuraavat sen virrankulutusta.

POW- ja HEAT-tolppiin verrattuna latauslaitteella on huomattavasti enemmän säätömahdollisuuksia. Latauslaite osaa päällä ja pois olon lisäksi sanoa, onko siihen kiinnitetty johtoa ja kulkeeko johdon läpi virtaa. Tämän vuoksi latauslaitteella on neljä erilaista tilaa:

1. Pois päältä
2. Päällä, mutta ei johtoa kiinni

3. Päällä, johto kiinni, mutta virta ei kulje
4. Päällä, johto kiinni ja virta kulkee.



Kuva 7. Sähköauton latauslaite.

3 Kuormantasausongelman esittely ja skenaariot

Seuraavaksi määritellään algoritmi sekä esitellään tarkemmin ongelmaa, jonka kuormantasausalgoritmi pyrkii ratkaisemaan. Ongelmaa käydään ensin läpi yleisellä tasolla, jonka jälkeen käsitellään erilaisia skenaarioita, joissa algoritmia tullaan käyttämään.

3.1 Algoritmi

Algoritmi on menettelytapa tietynlaisen määritellyn tehtävän suorittamiseen. Jokaisen kunnollisen tietokoneohjelman taustalla on algoritmi, jota voidaan pitää ohjelman ideana. Algoritmeja käytetään ongelmien ratkaisemiseen. [Skiena 2008, s. 3.]

Eräs toinen algoritmin määritelmä on ohjelma, joka saa syötteenä arvoja, tekee arvoille ennalta määritellyt toimenpiteet ja tuottaa arvoja ulos. Algoritmi on siis eräänlainen toimenpiteiden ketju, joka muuntaa syötteen halutunlaiseksi tulosteeksi. Nämä ennalta määritellyt toimenpiteet toimivat tietokoneelle ohjeistuksena ja niiden avulla syötteestä muovataan halutunlainen lopputulema. Algoritmi on siis työkalu laskennallisten ongelmien ratkaisemiseen. Tietokoneohjelmia voidaan pitää algoritmeina, sillä ainoa vaatimus algoritmille on, että sen tulee sisältää tarkka ohjeistus tarvittavista toimenpiteistä. [Cormen *et al*. 2009, ss. 5-6.]

Algoritmia voidaan pitää oikeellisena, mikäli se kaikilla mahdollisilla syötteillä tuottaa oikeanlaisen tuloksen. Oikeellinen algoritmi siis ratkaisee annetun ongelman. Väärin toimiva algoritmi ei välttämättä osaa käsitellä kaikkia sille tulevia syötteitä tai se saattaa tuottaa vääränlaisen tuloksen. [Cormen *et al*. 2009, s. 6.]

Yksinkertainen esimerkki algoritmista on lajittelualgoritmi, joka lajittelee saadut syötteet pienimmästä suurimpaan. Algoritmi saa syötteenä esimerkiksi luvut {6, 3, 4, 1} ja tulostaa {1, 3, 4, 6}. [Cormen *et al*. 2009, s. 5.] Kyseinen lajittelualgoritmi toimii oikeellisesti, mikäli se jokaisella ajokerralla järjestää numerot pienimmästä suurimpaan ja tulostaa vastauksen. Tällöin voidaan sanoa, että algoritmi ratkaisee lajitteluongelman. Väärin toimiva algoritmi saattaisi esimerkiksi hukata osan numeroista tai tulostaa niitä väärässä järjestyksessä.

Tämän työn algoritmi on ohjelma, joka reagoi syötteenä tulevaan dataan ja sen pohjalta tekee tarvittavia toimenpiteitä. Oikeellinen algoritmi osaa tehdä tarvittavat laskut ja niiden pohjalta tehdä tilanteen vaatimat toimenpiteet. Väärin toimiessaan algoritmi esimerkiksi saattaisi tehdä virheitä laskuissa ja toteuttaa vääriä toimenpiteitä.

3.2 Kuormantasausongelman esittely yleisesti

Alun perin kuormantasausta on käytetty sähköntuotannossa tasaamaan piikkejä sähköntarpeessa. Sähköntarve vaihtelee kellonajan mukaan. Esimerkiksi öisin sähköntarve laskee, kun ihmiset nukkuvat ja nousee jälleen, kun ihmiset heräävät ja rupeavat käyttämään sähkölaitteita. Sähkövoimalat eivät pysty kuitenkaan reagoimaan kovin nopeasti sähkön-

tarpeen vaihteluun esimerkiksi sammuttamalla voimalaa. Tämän vuoksi käytetään kuormantasausta, jolloin sähkö säilötään silloin, kun siitä on ylitarjontaa, ja tämä säilötty sähkö käytetään kulutuspiikkien aikaan. [Membrey *et al*. 2012, s. 3.]

Tietokonemaailmassa kuormantasausta tarvitaan myös, sillä usein käytössä olevat resurssit ovat rajalliset. Ratkaisut ovat kuitenkin hieman erilaisia kuin sähköntuotannossa, sillä tietokoneiden suoritustehoa ei ole mahdollista säilöä hiljaisempina hetkinä ja käyttää tarvittaessa, kuten sähköä. Jos nettisivun liikennemäärät nousevat, on pakko hankkia lisää suoritustehoa, jotta jokaista käyttäjää voidaan palvella. [Membrey *et al*. 2012, s. 3.]

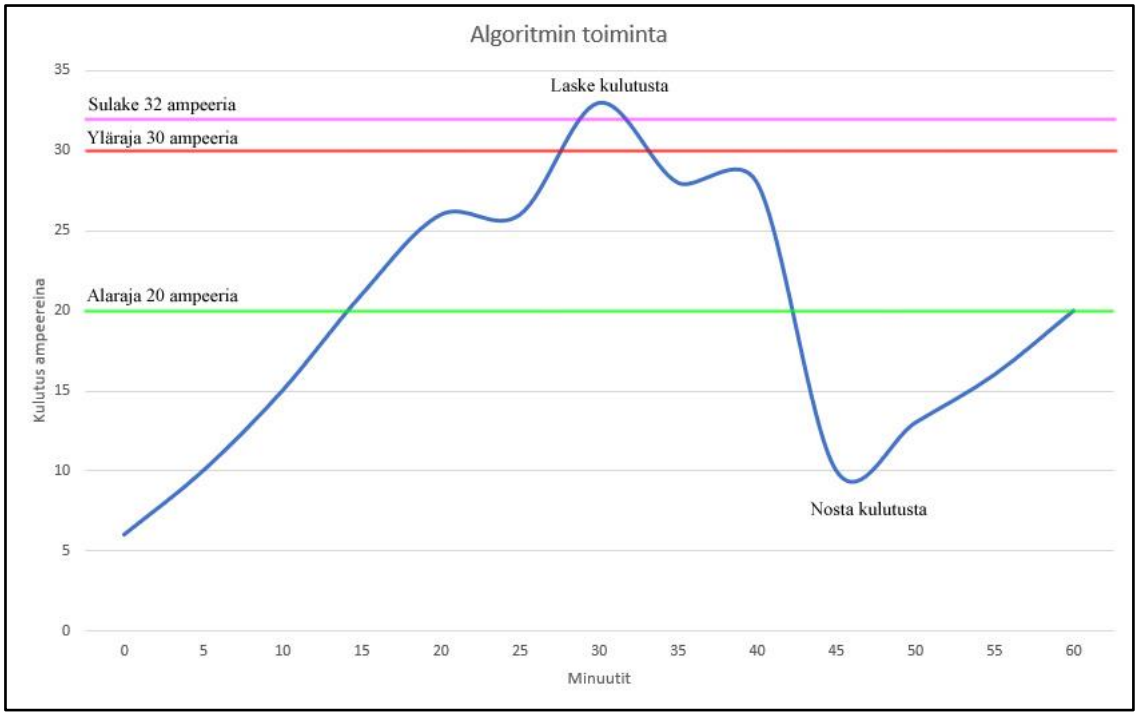
Kumpikaan näistä kuormantasaustavoista ei täysin sovi tämän työn tapaan tasata kuormaa, sillä ylimääräistä virtaa ei ole mahdollista säilöä talteen eikä myöskään ole mahdollista hankkia tehokkaampia laitteita. Kuormantasaus onkin enemmänkin rajallisten resurssien kanssa toimimista ja yritystä hyödyntää niitä mahdollisimman hyvin [Membrey *et al*. 2012, s. 3].

Olet varmasti joskus kokenut tai kuullut ongelmasta, jossa talosta palaa sulake, kun liikaa sähkölaitteita on päällä samanaikaisesti. Tämä on varsinkin vanhemmissa omakotitaloissa yleinen ongelma. Vastaavaan ongelmaan voidaan törmätä myös parkkipaikoilla, varsinkin jos osaa laitteista käytetään sähköautojen lataamiseen. Tässä työssä esiteltävä kuormantasausalgoritmi pyrkii ratkaisemaan edellä kuvatun ongelman.

Algoritmi kuuntelee vaiheen virrankulutusta ja toimii sen pohjalta. Algoritmin perustoiminnot voidaan jakaa kahteen osaan, joista ensimmäinen on vaiheen kuorman laskeminen. Kun virrankulutus nousee yli sallitun rajan, algoritmi pyrkii laskemaan kulutusta. Toinen perustoiminto on vaiheen kuorman nostaminen: jos kulutusta on varaa nostaa, niin sitä pyritään nostamaan.

Kuvassa 8 on esitelty algoritmin toimintaa esimerkkitilanteessa tunnin ajalta. Esimerkin sulake on 32 ampeeria. Yläraja, jonka yläpuolelle mentäessä algoritmi alkaa rajoittaa kulutusta, on asetettu 30 ampeeriin. Alaraja, jonka alle mentäessä algoritmi pyrkii nostamaan kulutusta, on asetettu 20 ampeeriin. Esimerkissä algoritmi laskee kulutusta heti kun se nousee yli ylärajan ja vastaavasti nostaa kulutusta hiljalleen, kun se alittaa alarajan. Laskemisen täytyy tapahtua kerralla kunnolla, jotta päästään nopeasti pois alueelta, joka saattaa saada sulakkeen palamaan. Kulutuksen nostamista kannattaa tehdä vähän kerrallaan, jotta kuorma ei nouse kerralla liikaa ja mene taas uudelleen ylärajan lähelle tai yläpuolelle.

Kulutus saattaa yhtäkkiä kasvaa tai vähentyä huomattavasti, jos esimerkiksi uusi sähkölaite lisätään verkkoon tai poistetaan verkosta. Tämän vuoksi on järkevää pitää sulakkeen ja ylärajan välillä tarpeeksi iso puskuri, jotta algoritmi ehtii toimimaan tarvittaessa. On vaikea arvioida sopivia ylä- ja alaraja arvoja etukäteen ilman testaamista käytännössä. Tämän vuoksi puskuriarvojen asettamista ja sopivia arvoja käsitellään myöhemmin tarkemmin toteutuksen esittelyn ja testauksen yhteydessä.



Kuva 8. Kuormantasausalgoritmin toiminta.

Käytännössä parkkipaikat koostuvat vaiheista ja jokaisella vaiheella on 0-x määrä POW-, HEAT- ja EVSE-tolppia. Kuorma lasketaan aina vaihe- tai sulakekohtaisesti. Algoritmi ohjailee näiden tolppien kulutusta ja päällä oloa komennoilla, joihin palataan toteutuksen yhteydessä tarkemmin. Algoritmin tulisi toimia oikeellisesti kaikilla erilaisilla tolppien kombinaatioilla ja määrillä.

Tiivistettynä tässä työssä kuormantasauksella siis tarkoitetaan vaiheelle kohdistuvan kuormituksen säätelyä niin, ettei se ylitä ennalta määritettyä ylärajaa, josta todennäköisesti seuraisi sulakkeen palaminen ja kulutuksen nostamista, mikäli se on mahdollista, jotta sähköautojen lataaminen olisi mahdollisimman optimaalista. Seuraavaksi esitellään tarkemmin sitä, miten rajoitettavat ja sammutettavat tolpat valitaan.

3.3 Toimenpiteet kulutuksen laskemiseen

Kulutusta täytyy laskea kerralla enemmän, jotta päästään ylärajasta tarpeeksi kauaksi. Tämän takia apuna käytetään arvoa, joka määrittää, kuinka paljon kulutuksen laskun pitää olla minimissään ylärajan alittamisen lisäksi.

Vaiheen kulutuksen laskemiseen on useita erilaisia tapoja. Ensimmäinen toimenpide, joka pyritään tekemään, on EVSE-tolppien kulutuksen rajoittaminen laskemalla tolppien virrankulutusta. Mikäli EVSE-tolppia on useampi, on rajoituksen tekemiseen useampia erilaisia vaihtoehtoja. On esimerkiksi mahdollista rajoittaa pisimpään päällä ollutta EVSE-tolppaa tai eniten ladannutta EVSE-tolppaa, jolloin vähemmän ladanneet autot saavat vielä jatkaa latausta suuremmalla teholla. Toinen mahdollinen tapa on rajoittaa jokaista tolppaa saman verran absoluuttisesti tai suhteellisesti kulutuksen mukaan.

Tämän työn ensimmäisen version rajoitustapana on kaikkien EVSE-tolppien kulutuksen laskeminen tasaisesti suhteessa kulutusmäärään. Ideana on laskea kaikkien latausteho, mutta eniten niiltä, jotka kuluttavat eniten. Myöhemmin algoritmiin on mahdollista lisätä toimintoja, jotka mahdollistavat jokaiselle vaiheelle ja kiinteistölle erilaiset tavat rajoitettavien EVSE-tolppien valintaan, esimerkiksi kiinteistön haltijan haluamalla tavalla.

Minimilaskumäärä = x
*MahdollisestiRajoitettavaKulutus = KaikkienEvsejenKulutus – (6 * EvseLkm)*
Prosentti = YksittäisenEvsenKulutus / MahdollisestiRajoitettavaKulutus
*YksittäisenTolpanRajoitusmäärä = Minimilaskumäärä * Prosentti*
Kaava 1. Tolppien rajoittaminen.

Kaava 1 esittää tavan, jolla rajaus lasketaan, mutta tarkemmin rajaustavan laskemiseen mennään toteutuksen ja skenaarioiden esittelyn yhteydessä. Yksi asiaa mutkistava tekijä on se, että EVSE-tolppaa ei voida laskea alle 6 ampeerin, joten matemaattisen kaavan täytyy ottaa huomioon vain yli 6 ampeerin menevät osuudet.

Toisena toimenpiteenä on EVSE-tolppien sammuttaminen kokonaan, mikäli niiden rajoittaminen ei riitä. Koska EVSE-tolppia voidaan rajoittaa minimissään 6 ampeeriin, täytyy EVSE-tolppa sammuttaa, mikäli suurempi pudotus kulutuksessa tarvitaan. EVSE-tolppia sammutetaan niin monta kuin tarvitaan, jotta päästään alle määritellyn tason, ja sammuttaminen aloitetaan pisimpään päällä olleesta.

Kolmas ja viimeinen mahdollinen toimenpide on POW- ja HEAT-tolppien sammuttaminen. Tämä toimenpide tehdään vain, jos EVSE-tolppia ei ole voitu sammuttaa tarpeeksi. Tämä tilanne voi tulla esimerkiksi, jos vaiheella ei ole yhtäkään EVSE-tolppaa tai jos niitä on vain yksi ja se on jo sammutettu, mutta kulutusta täytyy silti vielä laskea. Tolpat valitaan päälläoloajan perusteella, jolloin pisimpään päällä ollut sammutetaan ensimmäisenä. POW- ja HEAT-tolppia sammutetaan, kunnes on päästy alle määritellyn tason. Seuraavaksi kerrotaan, miten kulutuksen nosto tapahtuu.

3.4 Toimenpiteet kulutuksen nostamiseen

Kulutuksen nostaminen on mahdollista käytännössä vain EVSE-tolpilla, sillä POW- ja HEAT-tolppien uudelleen päälle laittaminen on haastavampi toteuttaa. Tämä johtuu esimerkiksi siitä, että POW- ja HEAT-tolpilla ei ole tietoa siitä, onko niissä johtoa kiinni. Olisi esimerkiksi täysin mahdollista, että julkisella parkkipaikalla POW- tai HEAT-tolpan sammuttamisen välissä paikalle parkkeerannut auto ehtisi vaihtua, jolloin lämmitys käynnistyisi auton omistajan tahtomatta väärällä hetkellä.

Mikäli vaiheen kulutusta on mahdollista nostaa, niin ensimmäinen toimenpide, joka pyritään tekemään, on algoritmin toimesta sammutettujen EVSE-tolppien uudelleenkäynnistäminen. Tämä toimenpide vaatii, että vaiheella on vähintään 6 ampeeria tilaa kulutuksessa, sillä EVSE-tolpan kulutus on minimissään 6 ampeeria. Uudelleenkäynnistetty EVSE-tolppa asetetaan ja rajoitetaan 6 ampeerin lataustehoon ennen uudelleenkäynnistämistä, jotta kulutus ei nouse liian paljoa kerralla.

Mikäli vaiheella ei ole sammutettuja EVSE-tolppia, niin aikaisemmin rajoitettujen EVSE-tolppien rajoituksia voidaan poistaa lataustehoa nostamalla. On hyvä huomioida, että EVSE-tolpalle asetettu maksimilatausvirta ei ole aina sama kuin auton todellisuudessa käyttämä latausvirta, sillä auto säätelee itsekin latausvirtaansa. Tästä saattaa seurata se, että vaikka lataustehoa nostaisi, niin virrankulutus ei välttämättä nouse.

Niin EVSE-tolppien lataustehon rajoittamiseen kuin myös rajoitusten poistamiseen on useita erilaisia tapoja. Esimerkiksi voidaan poistaa rajoituksia kaikilta tasaisesti tai vähiten ladanneelta kokonaan ja muilta ei ollenkaan. Tärkeintä rajoituksen poistossa on kuitenkin se, ettei sitä poisteta liikaa kerralla, josta seuraisi kokonaiskuorman nouseminen ylärajan lähettyville tai ylitse. Myöhemmin toteutukseen on mahdollista lisätä muitakin tapoja rajoituksen poistamiseen, jolloin esimerkiksi kiinteistön omistajalla olisi mahdollista valita itse, miten rajoitusten poisto tapahtuu. Tässä työssä rajoitusten poistoon valittiin tapa, joka on idealtaan samanlainen rajoitusten asettamisen kanssa, eli rajoitusta poistetaan jokaiselta EVSE-tolpalta tasaisesti suhteessa tolpan rajoitusmäärään (kaava 2). Enemmän rajoitetuilta tolpilta poistetaan rajoitusta enemmän kuin vähemmän rajoitetuilta. Jotta vältyttäisiin liian suurilta kertanostoilta, on apuna käytettävä arvoa, joka kertoo, kuinka paljon virrankulutusta voidaan yhdellä kerralla nostaa kokonaisuudessa.

Kokonaisnostomäärä = x
Prosentti = YksittäisenEvsenRajoitus / KaikkienEvsejenRajoitukset
*YksittäisenTolpanNostomäärä = Kokonaisnostomäärä * Prosentti*
Kaava 2. Kaava rajoitusten poistoon.

Rajoitusten poistoa käsitellään tarkemmin skenaarioiden yhteydessä kohdassa 3.6. Seuraavaksi esitellään algoritmin oikeellinen toiminta.

3.5 Oikeellinen toiminta

Seuraavaksi määritetään algoritmin oikeellinen toiminta. Kuormantasausalgoritmin voidaan sanoa toimivan oikeellisesti, kun se tekee seuraavat toimenpiteet kaikissa mahdollisissa erilaisissa skenaarioissa.

1. Laskee kulutusta, kun yläraja ylittyy:
    a. Ensisijainen toimenpide EVSE-tolppien kulutuksen rajoittaminen.

b. EVSE-tolppien sammuttaminen, mikäli EVSE-tolppia ei voida rajoittaa tarpeeksi.
c. Mikäli EVSE-tolppia ei ole tai niitä ei voida sammuttaa tarpeeksi, niin POW- ja HEAT-tolppien sammuttaminen.

2. Nostaa kulutusta, kun alaraja alittuu:
   a. Ensisijaisena toimenpiteenä EVSE-tolppien uudelleenkäynnistäminen, mikäli sammutettuja EVSE-tolppia on.
   b. Jos sammutettuja EVSE-tolppia ei ole, niin vähennetään EVSE-tolppien rajoituksia.

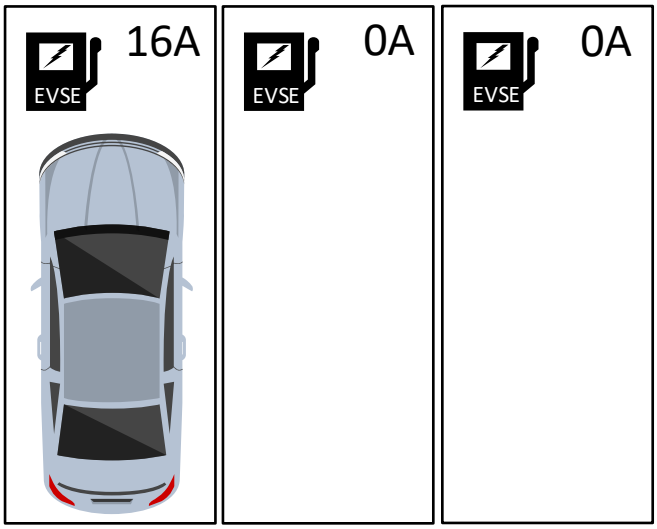
Seuraavaksi käsitellään erilaisia mahdollisia skenaarioita tarkemmin ja kerrotaan miten algoritmin tulisi niissä toimia, jotta se toimii oikeellisesti.

3.6 Skenaariot

Seuraavaksi esitellään algoritmin oikeellista toimintaa erilaisissa mahdollisissa skenaariossa esimerkkiparkkipaikkojen avulla.

3.6.1 Skenaario 1

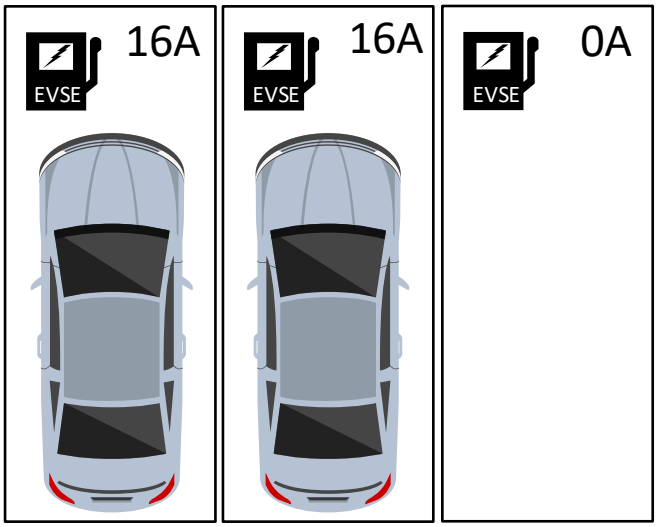
Ensimmäisessä skenaariossa esimerkkiparkkipaikalla on kolme EVSE-tolppaa, joiden maksimilatausvirrat ovat 16 ampeeria, 16 ampeeria ja 10 ampeeria. Nämä kolme EVSE-tolppaa on liitettynä samaan vaiheeseen, jonka sulake on 32 ampeeria. Algoritmin yläraja, joka ylittyessä kulutusta ruvetaan laskemaan, on 30 ampeeria. Ylärajan ylittyessä tehdään vähintään 6 ampeerin vähennys virrankulutuksessa. Alaraja on asetettu 26 ampeeriin, jonka alapuolelle mentäessä algoritmi poistaa tolpilta mahdollisia rajoituksia. Maksiminosto, joka voidaan tehdä alarajan alapuolelle mentäessä, on 3 ampeeria.



Kuva 9. Aloitustilanne parkkipaikalla.

Kuvassa 9 on kuvattu parkkipaikan alkutilanne. Autoja on latauksessa yksi, joka lataa täydellä 16 ampeerin latausvirralla. Kaksi muuta paikkaa on tyhjinä. Virtaa on siis vapaana 14 ampeeria ylärajaan ja 16 ampeeria sulakkeeseen verrattuna.

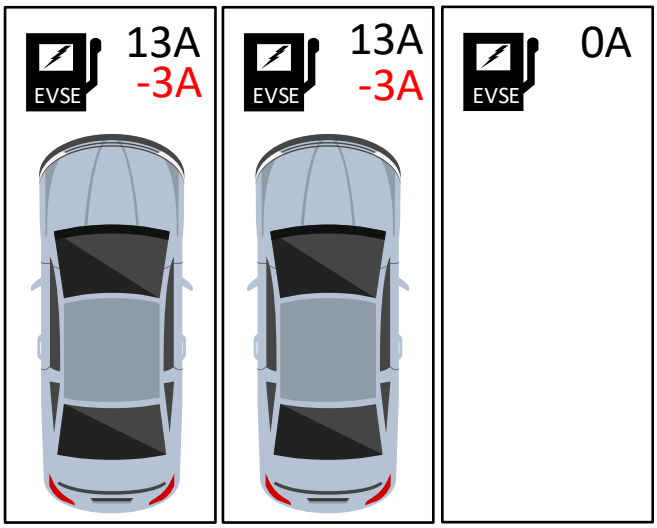
Kuvassa 10 parkkipaikalle saapuu toinen auto ja kokonaisvirrankulutus nousee 16 ampeerista 32 ampeeriin. Tämä tarkoittaa sitä, että yläraja ylitetään 2 ampeerilla.



Kuva 10. Toinen auto saapuu lataukseen ja virrankulutus nousee 16 ampeerista 32 ampeeriin, joka ylittää ylärajan.

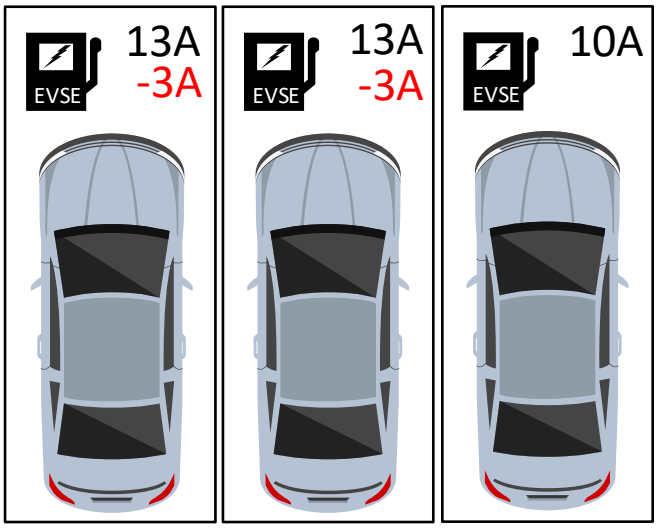
Kuvassa 11 näytetään tilanne virrankulutuksen rajoittamisen jälkeen. Molempien tolppien virrankulutusta lasketaan 3 ampeeria, joka tarkoittaa, että kokonaisvirrankulutus laskee 32 ampeerista 26 ampeeriin. Rajoitusmäärät lasketaan ensin selvittämällä, kuinka paljon kulutusta tarvitsee yhteensä laskea. Kulutuksen tulee olla vähintään 26 ampeeria rajoitusten jälkeen, koska kulutusta pyritään vähintään laskemaan 4 ampeeria ylärajasta. Ylärajan yläpuolella on 2 ampeeria, joka tarkoittaa, että kulutuksen tulee yhteensä laskea 6 ampeeria. Molempia tolppia on mahdollista rajata 10 ampeeria, koska 6 ampeeria on tolppien minimivirta, eli rajoituksia on mahdollista tehdä yhteensä 20 ampeeria. Tämä riittää hyvin 6 ampeerin rajoituksiin, joten tolppia ei tarvitse sammuttaa.

Tämän jälkeen täytyy laskea tolppakohtaiset rajoitukset. Ensin kokonaisvirrankulutuksesta vähennetään 6 ampeeria jokaista tolppaa kohden, sillä tätä osuutta ei voida rajoittaa. Saadaan 20 ampeeria, josta lasketaan kuinka suuren osan tolppa aiheuttaa. Vastaavasti myös tolppien kulutuksista vähennetään 6 ampeeria, eli molemmat tolpat kuluttavat 10 ampeeria. Kun 10 ampeeria jaetaan 20 ampeerista, saadaan prosenttiluvut, jotka kertovat kuinka paljon tolpan kulutus on kokonaismäärästä. Aluksi laskettiin, että kulutusta tarvitsee laskea 6 ampeeria, jotta päästään sopivasti alle ylärajan. Tämän vuoksi prosenttiluvulla kerrotaan 6 ampeeria, josta saadaan tolppakohtainen rajoitusmäärä. Molemmille tolpille prosenttiosuus on 50%, eli molempia tolppia rajoitetaan 3 ampeeria (kuva 11).



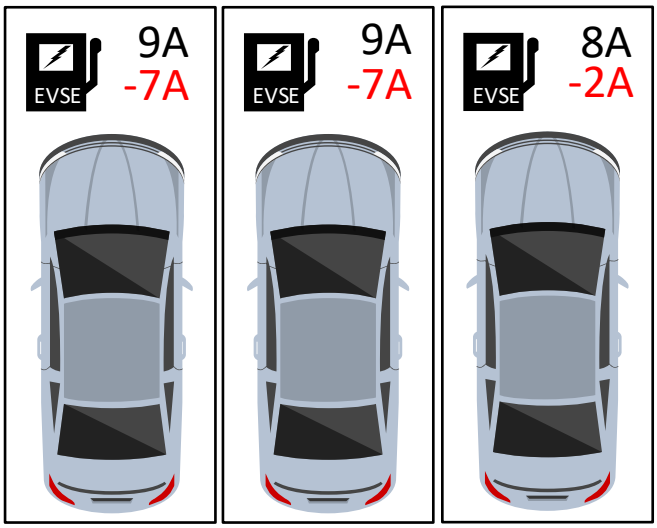
Kuva 11. Algoritmi rajoittaa autojen latausvirtoja, jotta päästään alle ylärajan.

Kuvassa 12 parkkipaikalle saapuu kolmas auto ja alkaa lataamaan 10 ampeerin virralla. Tästä seuraa se, että kokonaisvirrankulutus nousee 36 ampeeriin, eli ylittää ylärajan.



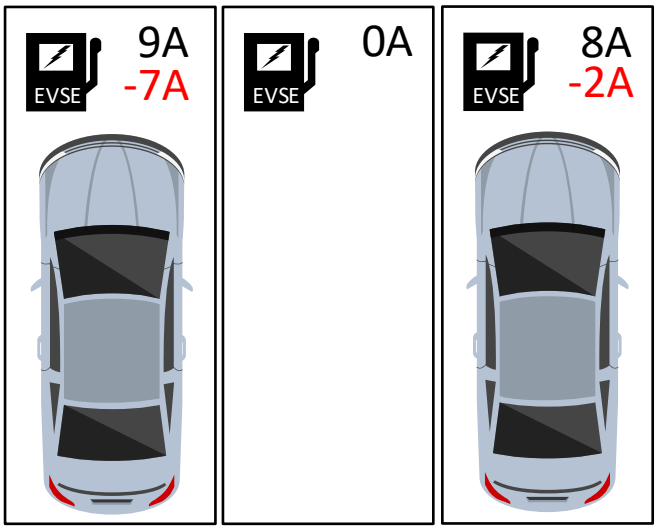
Kuva 12. Uusi auto tulee lataukseen ja kokonaisvirrankulutus nousee 36 ampeeriin, eli yli ylärajan.

Kuvassa 13 latauslaitteiden kulutusta on rajoitettu vastaavanlaisesti kuin aikaisemminkin. Ensin lasketaan, kuinka paljon kulutusta tarvitsee laskea, jotta päästään takaisin halutulle tasolle. Tässä tapauksessa kulutusta halutaan laskea 10 ampeeria (36 - (32 - 6), eli kokonaiskulutus - (sulake - minimilaskumäärä)). Tämän jälkeen lasketaan, kuinka suuren prosenttiosuuden kukin latauslaite käyttää. Ensimmäinen ja toinen latauslaite käyttävät noin 39% ja kolmas 22%. Tästä saadaan kahdelle ensimmäiselle latauslaitteelle 4 ampeerin ja kolmannelle 2 ampeerin rajoitus. Nämä rajoitukset lisätään mahdollisten vanhojen rajoitusten päälle. Yhteensä kulutus laskee siis 10 ampeeria, joka tarkoittaa, että kokonaiskulutus laskee 26 ampeeriin.



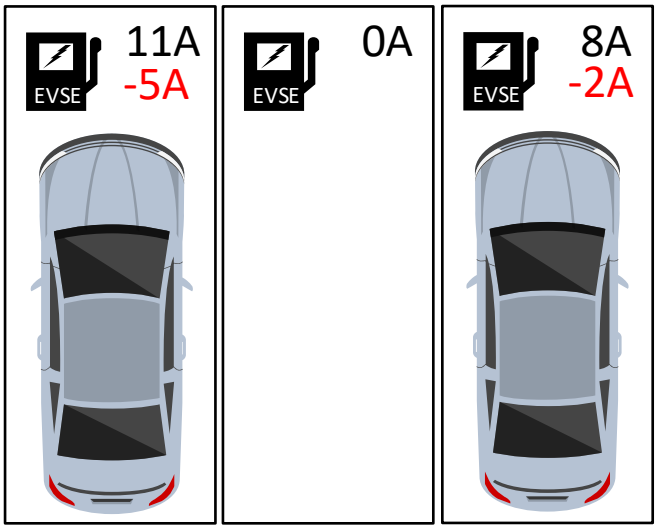
Kuva 13. Autoja rajoitetaan, jolloin kokonaiskulutus laskee 36 ampeerista 26 ampeeriin.

Kuvassa 14 keskimmäinen auto lähtee latauksesta, jonka jälkeen kokonaiskulutus putoaa 26 ampeerista 17 ampeeriin. Tämä alittaa alarajan reilusti, joten tolppien rajoituksia voidaan poistaa.



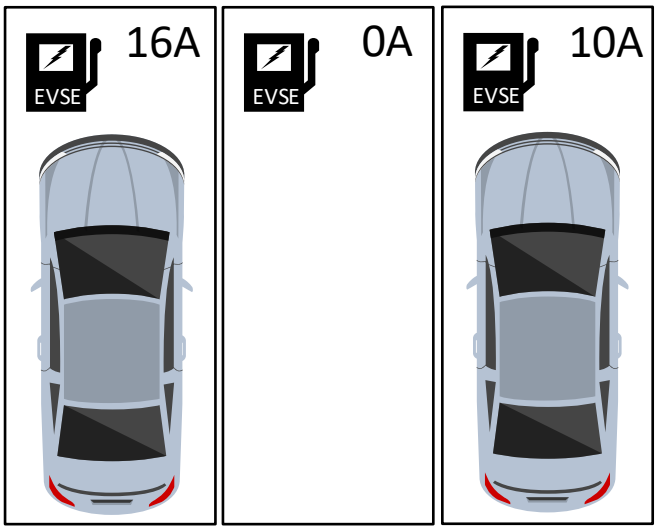
Kuva 14. Auto lähtee latauksesta ja kokonaiskulutus putoaa 17 ampeeriin.

Kuvassa 15 autojen rajoituksia poistetaan. Koko vaiheen maksiminostomäärä, joka latauslaitteille voidaan tehdä alarajan alittuessa, on 3 ampeeria. Rajoitusten poisto tehdään laskemalla latauslaitekohtaisesti, kuinka paljon rajoitusta poistetaan. Yhteensä rajoituksia on 9 ampeeria, joista 7 ampeeria on ensimmäisellä latauslaitteella ja 2 ampeeria kolmannella. Ensimmäisellä latauslaitteella on siis noin 78% (7/9) rajoituksista ja toisella noin 22% (2/9). Tämän jälkeen prosenttiosuudet kerrotaan maksiminostomäärällä, jolloin saadaan tietää, kuinka paljon yksittäisen latauslaitteen rajoituksia poistetaan. Ensimmäisen latauslaitteen rajoituksia poistetaan 2 ampeeria, toisen rajoituksia ei poisteta vielä yhtään. Luvut pyöristetään alaspäin, jotta ei tehdä liian isoja nostoja.



Kuva 15. Rajoituksia poistetaan latauslaitteilta.

Kuvassa 16 on lopputilanne, kun kaikki rajoitukset on hiljalleen poistettu molemmilta tolpilta ja kokonaiskulutus on noussut 26 ampeeriin.

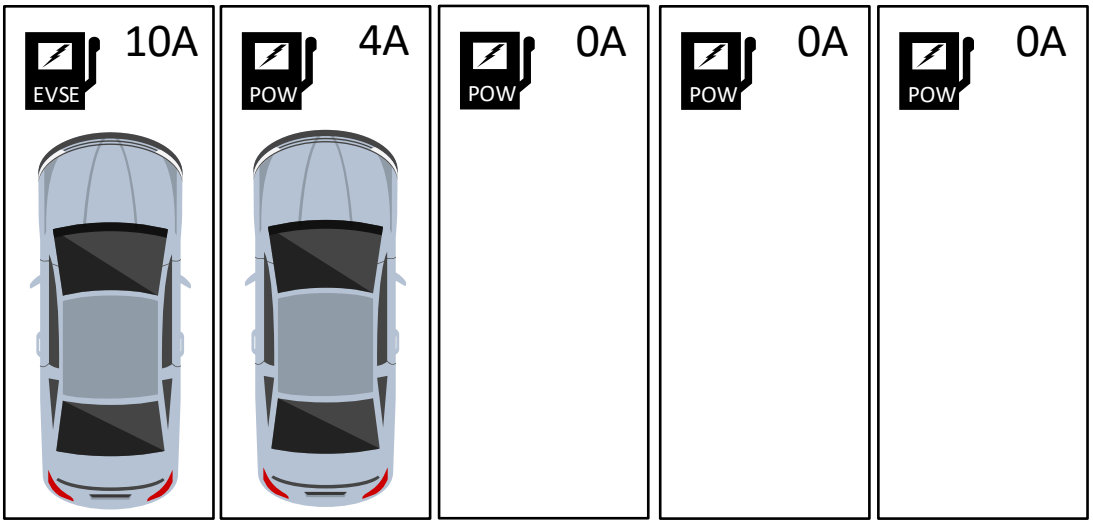


Kuva 16. Rajoitukset poistettu kokonaan molemmilta latauslaitteilta.

3.6.2 Skenaario 2

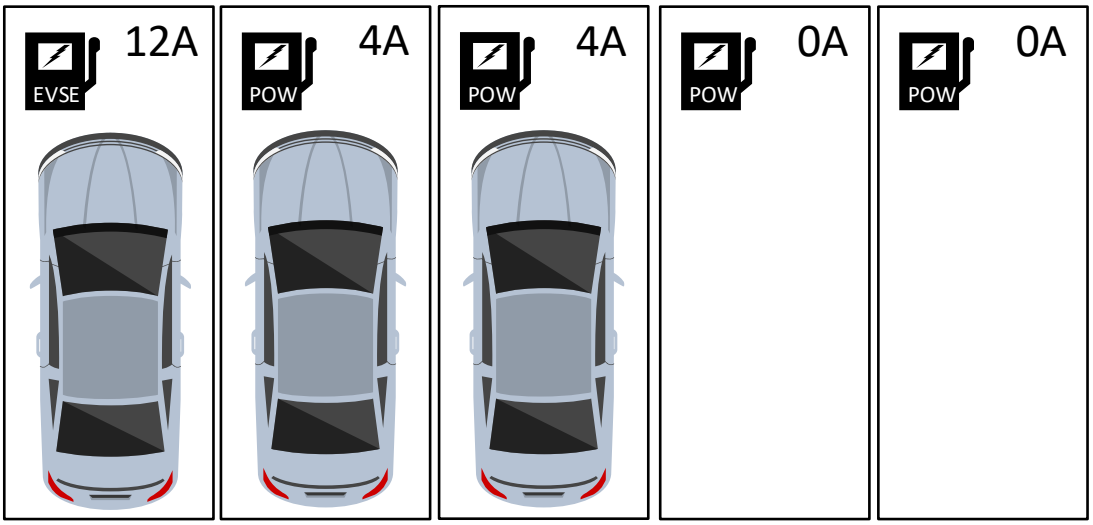
Toisessa skenaariossa on mallinnettu parkkipaikka, jossa on yksi EVSE-tolppa, jonka maksimivirta on 12 ampeeria, ja 4 POW-tolppaa. Tolpat ovat samassa vaiheessa ja vaiheen sulake on 16 ampeeria. Algoritmin yläraja on asetettu 15 ampeerin ja minimilaskumäärä 2 ampeeriin. Alaraja on 12 ampeeria ja maksiminostomäärä 2 ampeeria.

Kuvassa 17 on kuvattu skenaarion alkutilanne. Parkkipaikalla on yksi auto EVSE-tolpassa ja toinen auto POW-tolpassa. Vaikka EVSE-tolpan maksimivirta on 12 ampeeria, tässä tilanteessa auto käyttääkin vain 10 ampeerin virtaa. Vaiheen kokonaiskulutus on 14 ampeeria.



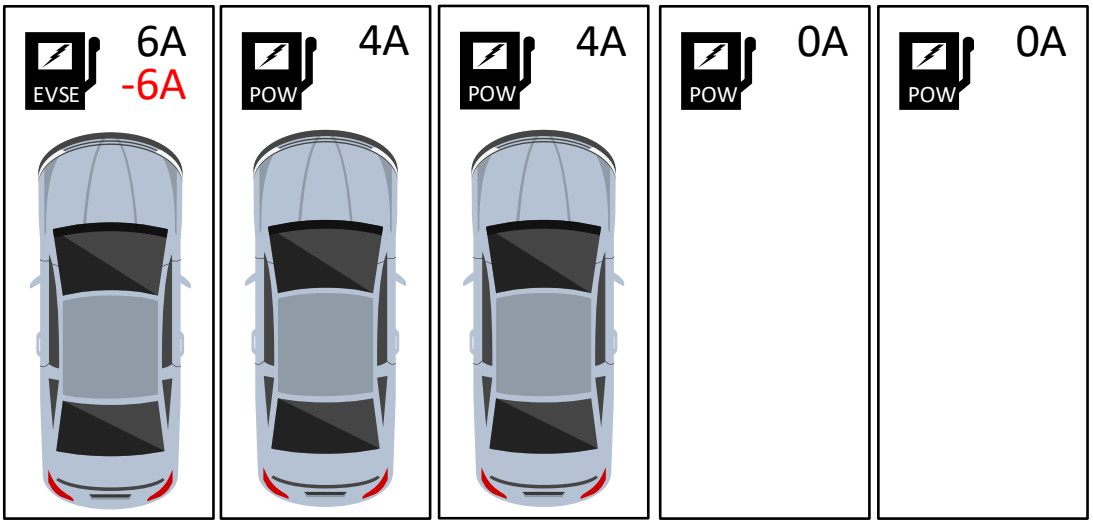
Kuva 17. Skenaarion aloitustilanne, eli yksi auto EVSE-tolpassa ja toinen POW-tolpassa. Vaiheen kokonaiskulutus on 14 ampeeria.

Kuvassa 18 EVSE-tolpan latausvirta nousee 10 ampeerista 12 ampeeriin. Samalla parkkipaikalle kolmanteen POW-tolppaan tulee auto lämmitykseen. Näin ollen vaiheen kokonaiskulutus nousee 20 ampeeriin, joka tarkoittaa sitä, että kulutusta tulee rajoittaa.



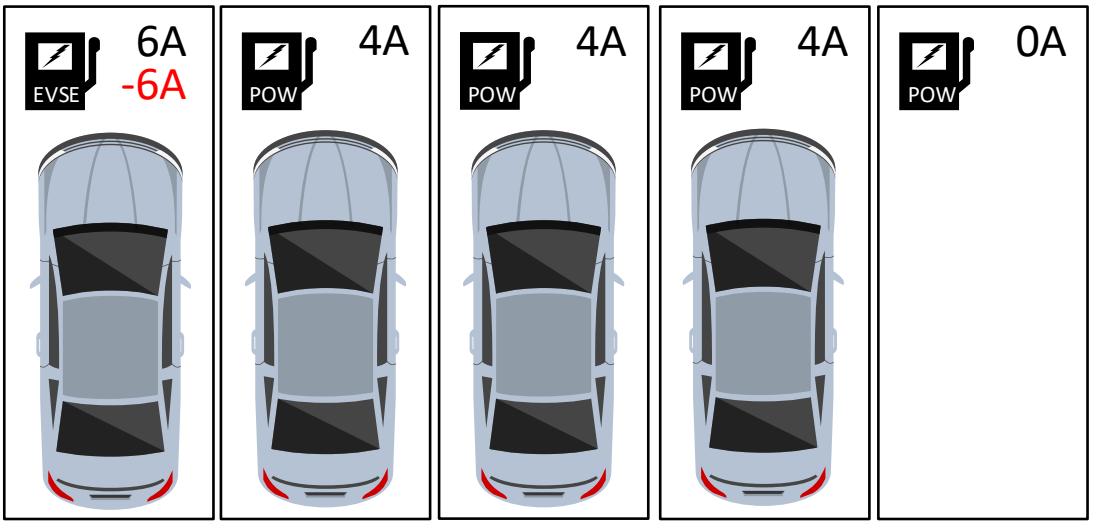
Kuva 18. Kolmas auto saapuu parkkipaikalle ja EVSE-tolpan kulutus nousee. Tästä seuraa kokonaiskulutuksen nouseminen 20 ampeeriin.

Rajoituksia tulee tehdä, koska kulutus on noussut ylärajan yläpuolelle. Ensin lasketaan, kuinka paljon kulutusta tulee laskea. Kulutusta tulee laskea yhteensä 6 ampeeria, koska 20 - (16 - 2) = 6 (kokonaiskulutus - (sulake - minimilaskumäärä). Koska vaiheella on EVSE-tolppa ja sitä voidaan rajoittaa riittävästi, rajoitetaan EVSE-tolppaa 6 ampeeria. Tästä seuraa se, että kokonaiskulutus laskee 14 ampeeriin (kuva 19).



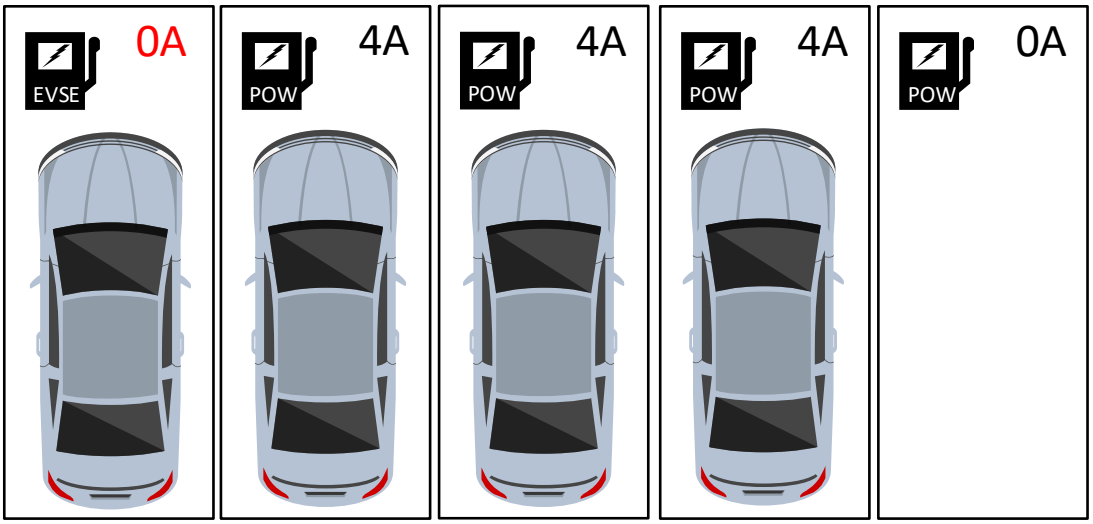
Kuva 19. Rajoitetaan EVSE-tolppaa. Kokonaiskulutus laskee 14 ampeeriin.

Neljäs auto saapuu parkkipaikalle (kuva 20) ja kokonaiskulutus nousee 18 ampeeriin.



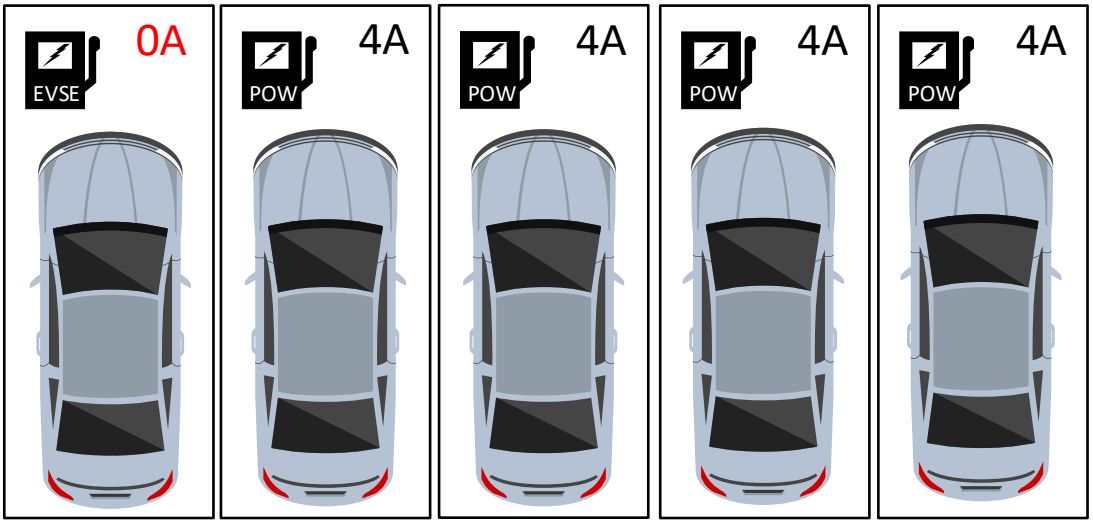
Kuva 20. Neljäs auto saapuu parkkipaikalle ja kokonaiskulutus nousee 18 ampeeriin.

Koska kokonaiskulutus on noussut 18 ampeeriin, kulutusta tulee laskea vähintään 4 ampeeria. EVSE-tolppaa ei voida rajoittaa enempää, sillä 6 ampeeria on EVSE-tolpan minimilatausvirta. Tästä seuraa se, että EVSE-tolppa sammutetaan (kuva 21). Kulutus laskee 12 ampeeriin.



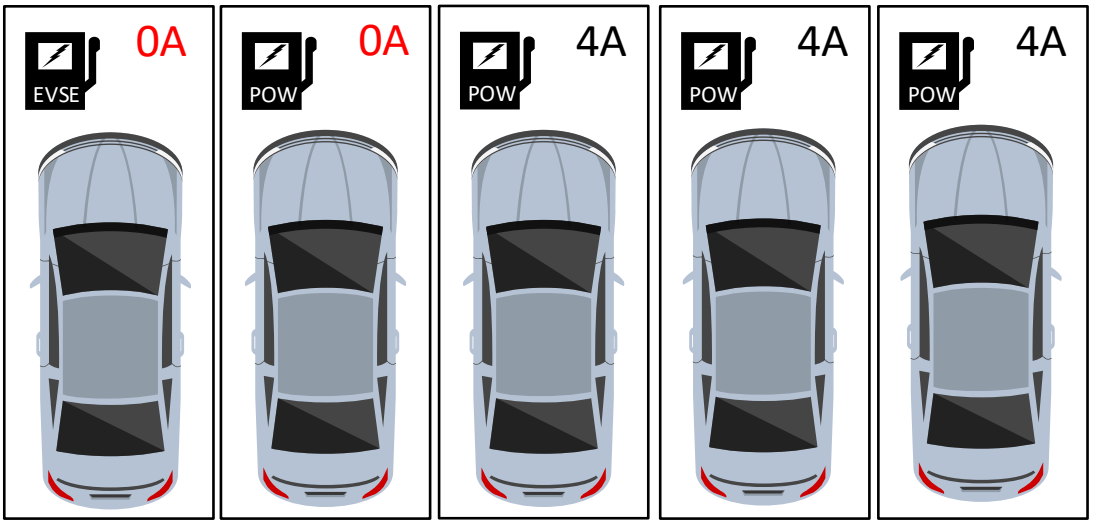
Kuva 21. EVSE-tolppa sammutetaan, jotta päästään alle 14 ampeerin.

Viides auto saapuu parkkipaikalle ja kulutus nousee 16 ampeeriin (kuva 22). Tästä seuraa se, että kulutusta on jälleen laskettava.



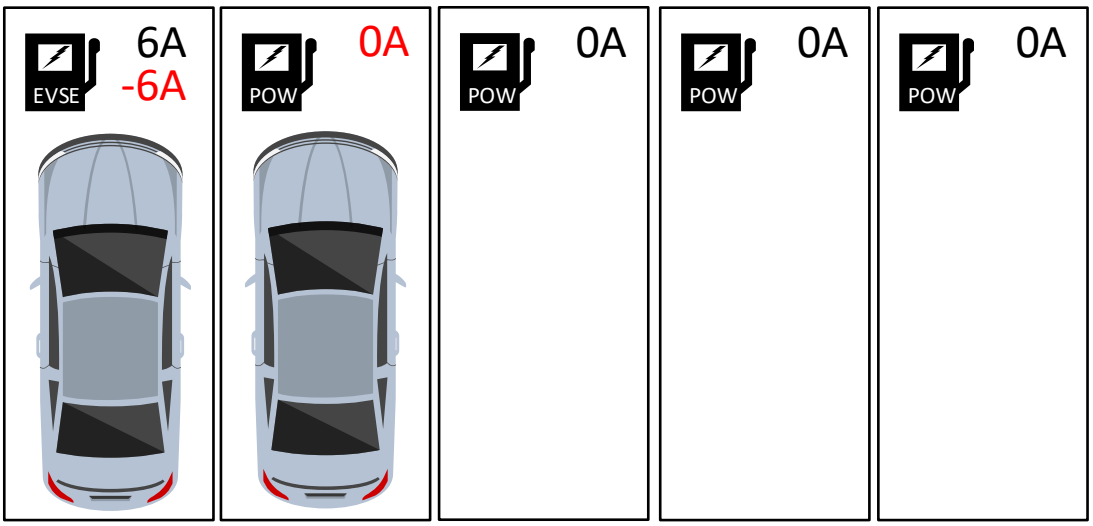
Kuva 22. Viides auto saapuu parkkipaikalle ja kulutus nousee 16 ampeeriin.

Viidennen auton saavuttua kulutusta on laskettava, jotta sulake ei pala, kun kulutus nousee 16 ampeeriin. Koska EVSE-tolppa on jo aikaisemmin sammutettu, niin jäljelle jää vain POW-tolppien sammuttaminen. Pisimpään päällä ollut POW-tolppa sammutetaan, jolloin kulutus laskee 12 ampeeriin (kuva 23).



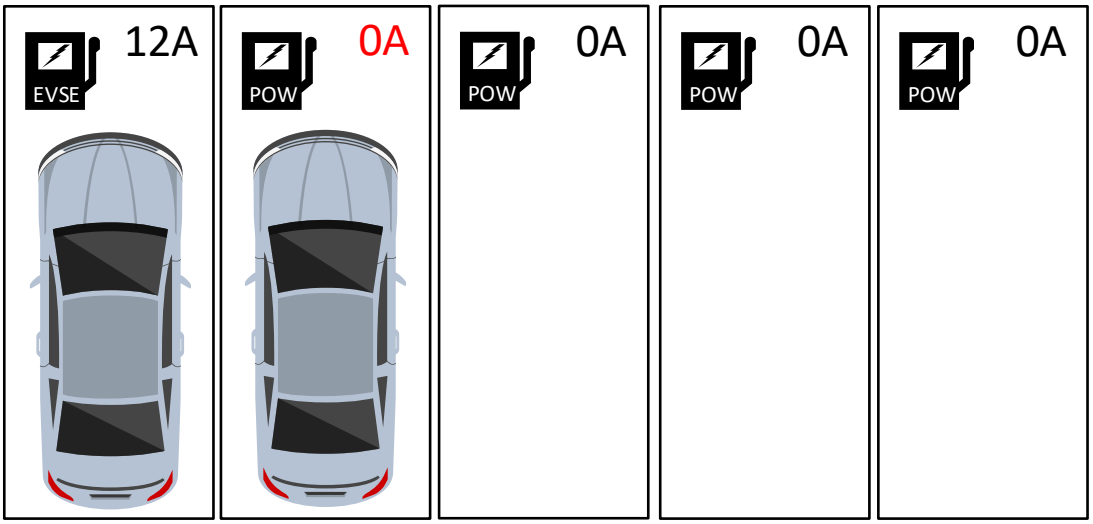
Kuva 23. Algoritmi sammuttaa POW-tolpan, jotta kulutusta saadaan laskettua.

Kuvassa 24 kolme autoa oikealta katsottuna lähtevät parkkipaikalta ja jäljelle jäävät EVSE-tolpan auto ja viereinen, aikaisemmin sammutettu POW-tolpan auto. Tästä seuraa se, että vaiheen virrankulutus on 0 ampeeria, koska molemmat tolpat ovat sammutettuina. Algoritmi nostaa virrankulutusta käynnistämällä EVSE-tolpan uudelleen ja asettamalla sen virrankulutuksen 6 ampeeriin. POW-tolppia ei käynnistetä uudelleen.



Kuva 24. Kolme autoa lähtee, joten EVSE-tolppa voidaan käynnistää uudelleen.

Kuvassa 25 on kuvattu lopputilanne, jossa algoritmi on poistanut EVSE-tolpan rajoitukset kokonaan ja virrankulutus on noussut maksimiin 12 ampeeriin. Kuvan 24 ja kuvan 25 välillä on useampi välivaihe, jossa algoritmi hiljalleen poistaa EVSE-tolpan rajoitusta, sillä maksiminostomäärä on asetettu 2 ampeeriin. POW-tolppa pysyy sammutettuna, sillä niitä ei käynnistetä uudelleen.



Kuva 25. Lopulta algoritmi poistaa EVSE-tolpan rajoitukset kokonaan.

4 Kuormantasausalgoritmin toteutus

Seuraavaksi käsitellään algoritmin toteutusta. Aluksi kohdassa 4.1 kerrotaan tarkemmin käytetyistä tekniikoista, kuten MQTT-protokollasta. Tämän jälkeen kohdassa 4.2 käsitellään tolppien ja algoritmin välistä viestiliikennettä. Viestiliikenteen kautta tuleva data tulee voida säilöä jotenkin algoritmin käytettäväksi, minkä vuoksi kohdissa 4.3 ja 4.4 käsitellään tietokanta- ja muistiratkaisuja. Kohdassa 4.5 esitellään muuttujat, joiden avulla voidaan säädellä tarkemmin algoritmin toimintaa. Kohdassa 4.6 esitellään algoritmin toteutusta vuokaavioiden avulla. Lopuksi kohdassa 4.7 käydään yhteenveto koko ratkaisun arkkitehtuurista.

4.1 Käytetyt tekniikat

Seuraavaksi esitellään olennaisia tekniikoita ja palveluita algoritmin ja tolppien toteutuksissa. Algoritmi itsessään on koodattu C#:lla Microsoftin Visual Studiossa.

4.1.1 Azure ja Azure Webjobit

Pilvilaskennalla tarkoitetaan laskentaresursseja ja palveluita, joita tarjotaan internetin kautta. Palvelut voidaan jakaa kolmeen tasoon: infrastruktuuri palveluna (IaaS), sovellusalusta palveluna (PaaS) ja ohjelmisto palveluna (SaaS). IaaS on alin taso ja se tarjoaa käyttäjälle serverin, tallennustilan ja verkkoinfrastruktuurin, johon käyttäjä voi itse rakentaa haluamansa palvelun. PaaS-taso sallii käyttäjän itse rakentaa applikaatioita ilman, että tarvitsee miettiä infrastruktuuria, eli palveluntarjoaja tarjoaa valmiita alustoja, jotka käyttäjä voi helposti ottaa käyttöönsä. SaaS-taso tarjoaa käyttäjälle valmiita ohjelmia käytettäväksi, esimerkki tällaisesta palvelusta on sähköpostiohjelmat. SaaS-tasolla palveluntarjoaja vastaa koko ohjelmistosta. [Webber-Cross 2014, s. 9-10.]

Yksi pilvilaskentaa tarjoava palvelu on Microsoftin Azure, joka tarjoaa lukuisia erilaisia palveluita käyttäjilleen IaaS- ja PaaS-tasoilla. Alun perin Azure julkistettiin vuonna 2008 ja vuodesta 2010 alkaen se on ollut kaupallisesti saatavissa kaikille. Julkaisusta lähtien palveluiden ja toimintojen määrä on kasvanut. Nykyisin Azuresta on saatavilla palveluita tietokannoista kasvojen tunnistamiseen. [Webber-Cross 2014, s. 10-11.]

WebJob on Azuren toiminto, joka mahdollistaa ohjelmien ajamisen web-applikaatioiden kanssa samassa kontekstissa. WebJobit voivat olla jatkuvia, jolloin ne ovat päällä jatkuvasti tai sitten niin, että ne ajetaan vain tiettynä kellonaikana, esimerkiksi joka yö kello kolme. [Microsoft 2018a.]

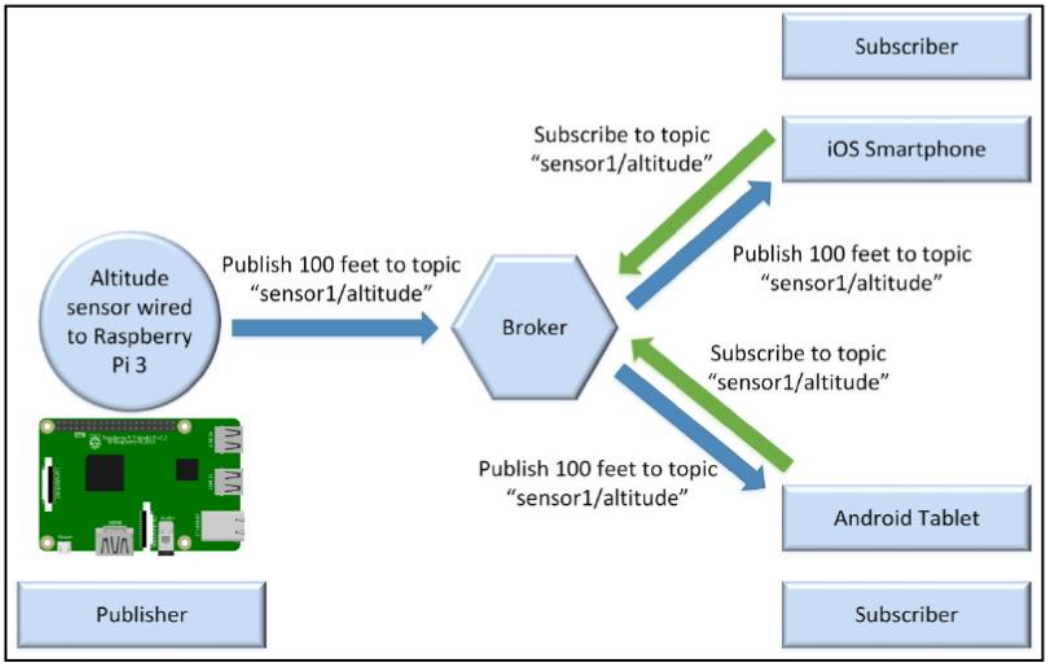
Tämän työn kuormantasausalgoritmi tulee pyörimään jatkuvana WebJobina Azuren pilvilaskentapalvelussa.

4.1.2 MQTT ja RabbitMQ

MQTT-protokolla (Message Queuing Telemetry Transport) on kevyt viestintäprotokolla, joka toimii julkaisija-tilaaja (publish-subscribe) mallilla. Tämä tarkoittaa sitä, että asiakas

julkaisee tietyntyyppisiä viestejä ja vain ne asiakkaat, jotka ovat kiinnostuneita tietyntyyppisistä viesteistä saavat viestit. Käytännössä viestejä julkaistaan tiettyyn aiheeseen (topic) ja viestejä voidaan lukea tilaamalla tämä aihe. Viestejä julkaiseva asiakas ei tiedä mitään muiden asiakkaiden, tilaajien tai julkaisijoiden, olemassaolosta. Julkaisija-tilaaja malli tarvitsee toimiakseen välittäjää (broker), eli serveriä, johon kaikki asiakkaat ottavat yhteyden. Julkaisija lähettää viestejä välittäjälle, joka välittää viestit tilaajille. [Hillar 2017, s. 9-11.]

Kuvassa 26 on avattu tarkemmin julkaisija-välittäjä-tilaaja mallia. Julkaiseva laite on Rasberry Pi 3 -laite, jossa on korkeussensori. Laite luo yhteyden välittäjään ja julkaisee ”sensor1/altitude” -aiheeseen viestejä. Android-tabletti ja iOS-älypuhelin ilmoittavat välittäjälle haluavansa tilata ”sensor1/altitude” -aiheen, jolloin laitteet saavat viestit välittäjältä, aina kun aiheeseen julkaistaan uusia viestejä. [Hillar 2017, s. 12.]



Kuva 26. Julkaisija-välittäjä-tilaaja malli [Hillar 2017, s. 12].

Keveytensä takia MQTT-protokolla on hyvä vaihtoehto esimerkiksi IoT-laitteisiin, sillä viestien julkaiseminen ei vie kovin paljon kaistaa ja toimii heikommallakin internetyhteydellä [Hillar 2017, s. 9-11]. Tässä työssä MQTT-protokollaa käytetään viestien välittämiseen molempiin suuntiin tolppien ja algoritmin välillä. Välittäjänä käytetään RabbitMQ-palvelua, joka on suosituin avoimen lähdekoodin välittäjä [RabbitMQ 2019].

4.2 Viestiliikenne

Tolppien ja algoritmin välinen viestiliikenne on elintärkeä osa koko ratkaisua. Ilman sitä algoritmi ei saisi tietoa virrankulutuksen ylittymisestä eikä voisi ohjata tolppia. Tolppien

ja algoritmin välinen viestiliikenne toimii siis molempiin suuntiin: tolpat ilmoittavat algoritmille virrankulutuksensa ja algoritmi lähettää tolpille viestejä virrankulutuksen nostosta tai rajoittamisesta. Tolpat ovat viestiyhteydessä yhteen algoritmiin ja algoritmi on viestiyhteydessä useampaan tolppaan. MQTT-protokollan keveyden takia tolppien viestiliikenne on erittäin skaalautuva, joka mahdollistaa tuhansien uusien tolppien lisäämisen ratkaisuun.

Viestiliikenne on jaettu kahteen alakohtaan, joista ensimmäisessä alakohdassa käsitellään tolpilta tulevaa viestiliikennettä ja toisessa alakohdassa algoritmin viestejä tolpille.

4.2.1 Tolppien viestiliikenne algoritmille

Tolppien viestiliikenne on runsasta, sillä tällä hetkellä yksittäinen tolppa lähettää kymmeniä viestejä 30 sekunnin välein. Viestit sisältävät paljon erilaista tietoa tolpan toiminnasta, mutta kuormantasausalgoritmin kannalta oleellisia viestejä ovat vain ne, jotka koskevat tolpan virrankulutusta. Tolppa lähettää virrankulutusviestin 30 sekunnin välein.

POW- ja HEAT-tolpilla virrankulutusviesti on nimeltään ja tyypiltään ”SENSOR”, eli sensoriviesti (viesti 1). Sensoriviesti sisältää virrankulutustiedon lisäksi myös muuta tietoa, kuten esimerkiksi energiankulutustietoja. Algoritmin kannalta oleellinen tieto viestistä on ”Current”-kenttä, joka ilmoittaa virrankulutuksen ampeereina.

SENSOR: {"Time":"2019-04-11T09:26:32","ENERGY":{"Total":244.491,"Yesterday":0.002,"Today":0.001,"Period":0,"Power":0,"Factor":0.00,"Voltage":0,"Current":0.000}}

Viesti 1. Esimerkki sensoriviestistä POW- ja HEAT-tolpalta.

EVSE-tolpilta saadut viestit eroavat POW- ja HEAT-tolppien viesteistä. EVSE-tolppa lähettää virrankulutustiedon viestinä, jonka nimi on ”amp” (viesti 2). Tämä viesti ei sisällä muuta tietoa kuin virrankulutustiedon milliampeereina. On myös hyvä huomioida, että toisin kuin sensoriviesti, amp-viesti ei sisällä aikaleimaa, eli tietoa siitä, koska viesti on lähetetty tolpalta.

amp: 29920

Viesti 2. Esimerkki ”amp”-viestistä EVSE-tolpalta.

Näiden viestien pohjalta algoritmi kerää tietoa tolppien virrankulutuksesta, jonka pohjalta algoritmi tekee tarvittavia toimenpiteitä. Myöhemmissä kohdissa esitellään tarkemmin, miten virrankulutustietoa säilötään ja käytetään.

4.2.2 Algoritmin viestiliikenne tolpille

Algoritmin tulee voida lähettää tolpille viestejä, jotta algoritmi voi toteuttaa tarvittavia toimia virrankulutuksen säätelemiseksi. Tolpille lähtevät viestit voidaan jakaa POW- ja HEAT-tolpille lähteviin viesteihin ja EVSE-tolppien viesteihin.

Kuten aikaisemmin kerrottiin, POW- ja HEAT-tolppia voidaan vain sammuttaa algoritmin toimesta. Tästä syystä niille lähetettäviä viestejä on vain yksi, joka sammuttaa tolpan (viesti 3). Tässä viestissä alussa tulee viestijonon nimi, jotta RabbitMQ osaa käsitellä viestin. Tämän jälkeen tulee tunniste, joka kertoo mihin tolppaan viestiä ollaan lähettämässä. Viestin tyyppi on ”POWER” ja komento ”OFF”, joka kertoo tolpalle, että virta pitää sammuttaa.

**viestijono*.*tunniste*.command.POWER: ”OFF”*

Viesti 3. POW- ja HEAT-tolpille lähetettävä sammutuskomento.

EVSE-tolpille lähtevä viestiliikenne on monimutkaisempaa, sillä EVSE-tolppia voidaan sammuttaa, käynnistää ja rajoittaa. Viesti koostuu viestijonon nimestä, tolpan tunnisteesta, ja komentotunnuksesta (viesti 4). Komentotunnuksia on kolmea erilaista. Ensimmäinen komento on ”FD”, joka tarkoittaa latauslaitteen sammuttamista. Toinen komento on ”FE”, joka tarkoittaa latauslaitteen käynnistämistä. Kolmas komento on ”SC”, joka asettaa latauslaitteelle maksimilatausvirran. Tämän komennon yhteydessä annetaan luku, joka asetetaan latauslaitteen maksimilatausvirraksi. Esimerkiksi ”SC 6” asettaa maksimilatausvirran 6 ampeeriin.

viestijono*.*tunniste*.cmnd.in.$*komentotunnus

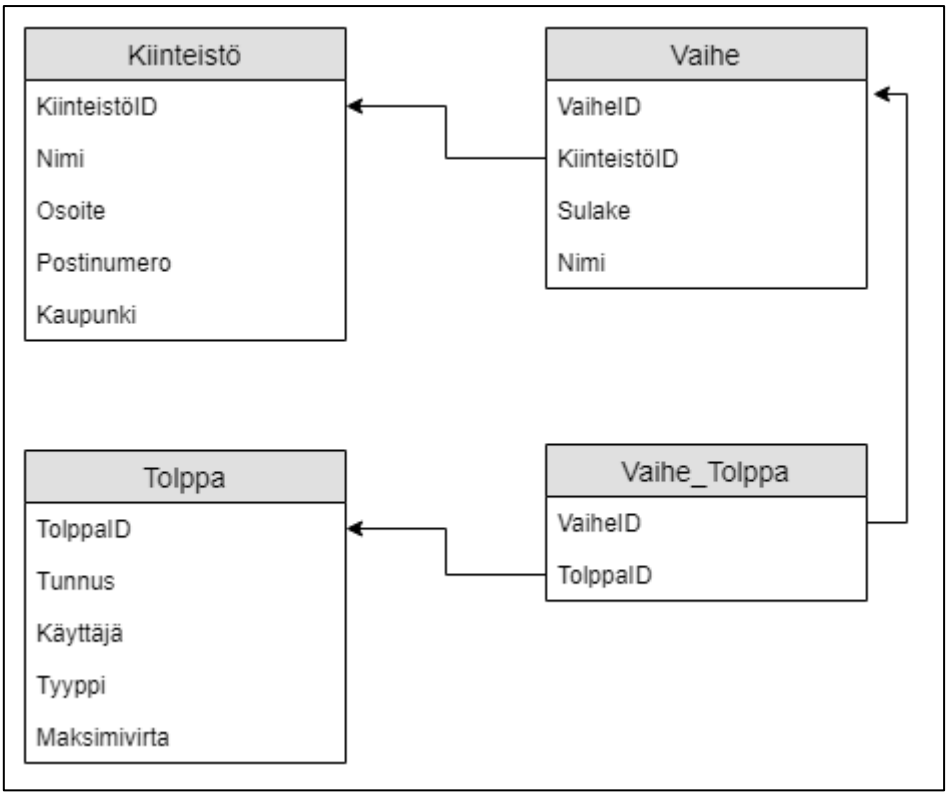
Viesti 4. EVSE-tolppien viestien runko.

Näiden viestien avulla tolppia ohjaillaan algoritmin toimesta.

4.3 Tietokantaratkaisu

Algoritmin eräs tärkeä tehtävä on tallentaa tolpilta tulevaa tietoa tietokantaan. Tämä on erillinen tehtävä kuormantasaukseen liittyen. Käytännössä algoritmi kuuntelee tolpilta tulevia viestejä ja kerää muistiin talteen niissä tulevat tiedot. Nämä tiedot tallennetaan tietokantaan niille tarkoitettuihin tauluihin 30 sekunnin välein. Tiedon kerääminen ja tietokantaan tallennus on suunniteltu mahdollisimman tehokkaaksi toiminnoksi ja toteutettu bulkkilisäyksenä (bulk insert). Jatkossa tätä tietoa voi kutsua raakatiedoksi, sillä sitä ei käsitellä mitenkään ja lisäys tehdään lisäämällä kantaan uusia rivejä. Näin erotamme sen algoritmille tärkeästä tiedosta. Raakatiedon osuutta ei käsitellä tässä työssä tarkasti, sillä se tapahtuu vain sivutoimintona kuormantasaustoiminnan ohessa.

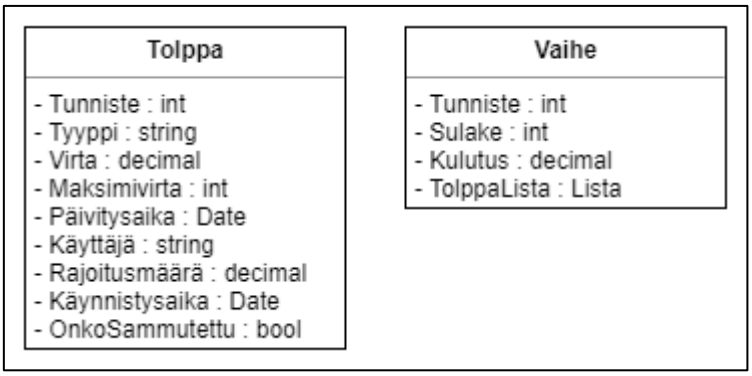
Tietokantaan on tallennettu rakenne myöskin tolpista, vaiheista ja kiinteistöistä (kuva 26). Tietokannassa rakenne menee niin, että kiinteistöllä voi olla useampi vaihe, vaiheella voi olla useampi tolppa ja tolpalla useampi vaihe. Vaiheella on tieto sulakkeen koosta. Tolpalla on tieto tolpan käyttäjästä, maksimivirrasta ja tyypistä, eli siitä, onko kyseessä EVSE-, POW- vai HEAT-tolppa. Näiden tietojen pohjalta muistiin rakennetaan tietorakenne, josta voidaan hakea vaiheelle kuuluvat tolpat tai tolpalle vaihe, johon se kuuluu. Muistiratkaisuun syvennytään tarkemmin seuraavassa kohdassa.



Kuva 26. Tietokantarakenne kiinteistölle, vaiheelle ja tolpalle.

4.4 Muistiratkaisu

Algoritmin toiminnan tehostamiseksi, vaiheiden ja tolppien tiedot tallennetaan muistiin sen sijasta, että tieto haettaisiin aina tietokannasta. Ohjelmaan on rakennettu oma luokka, joka toimii tietorakenteena tolpille ja vaiheille. Pohjimmiltaan tämä luokka koostuu tolppa- ja vaiheolioista (kuva 27). Vaiheella on tunniste, tieto sulakkeen koosta, tieto vaiheen kokonaisvirrankulutuksesta ja lista sille kuuluvista tolpista. Tolpalla on tunniste, tyyppi (EVSE/POW/HEAT), virta, maksimivirta, päivitysaika, käyttäjä, rajoitusmäärä, käynnistysaika ja tieto siitä onko tolppa sammutettu algoritmin toimesta. Päivitysaikaa päivitetään aina kun tolpalta tulee virrankulutusviesti.



Kuva 27. Tolppa- ja vaiheoliot.

Näiden olioiden pohjalta on rakennettu tietorakenne, jonne voidaan lisätä vaiheita ja tolppia vaiheelle. Tämän lisäksi voidaan myös tehdä hakuja vaiheen tunnisteella tai tolpan tunnisteella, jolloin palautuksena tulee aina koko vaihe. Tietorakenteessa on hyödynnetty C#-kirjastojen valmiita tietorakenteita, kuten esimerkiksi dictionary-tietorakennetta, joka koostuu avaimesta ja siihen liitetystä arvosta. Dictionary-tietorakenteesta avaimen perusteella hakeminen on erittäin nopeaa, lähellä nopeutta O(1) [Microsoft 2019]. Algoritmin tietorakenteessa käytetään kahta dictionary-tietorakennetta, josta ensimmäisessä on parina vaiheen tunniste ja vaiheolio, ja toisessa tolpan tunniste ja vaiheen tunniste. Tämä mahdollistaa hakujen tekemisen nopeasti sekä vaiheen että tolpan tunnisteella.

On täysin mahdollista, että tolppien ja vaiheiden muuttujien arvot muuttuvat tietokannassa. Esimerkiksi tolpan maksimivirta-arvo saattaa muuttua tietokannassa, jolloin tieto muutoksesta tulee myös saada algoritmille. Muuten voisi seurata tilanne, jossa virtaa nostetaan algoritmin toimesta korkeammaksi kuin laitteet oikeasti kestävät, mikä voisi mahdollisesti aiheuttaa vakavan virheen laitteiden toimintaan. Tästä syystä on tärkeää, että algoritmin tiedot ovat ajan tasalla. Toisaalta jokaisella kerralla, kun tolppien ja vaiheiden tietoja tarvitaan, uusimpien arvojen hakeminen tietokannasta on raskas ja hidas prosessi. Tästä syystä on tehty eräänlainen kompromissi, jossa vaiheiden ja tolppien tiedot haetaan tietokannasta muistiin aina samalla kun raakatietoa päivitetään tietokantaan, eli 30 sekunnin välein. Koska kaikkia algoritmin tarvitsemia tietoja ei saada suoraan tietokannasta, on tietorakenteeseen toteutettu päivitysfunktio, joka vertailee tietokannasta tullutta tietoa jo muistissa olleeseen tietoon ja poimii vain muutokset. Esimerkki tiedosta, jota tietokannasta ei saada, on tolpan päivitysaika, sillä algoritmi päivittää arvon muistiin, mutta sitä ei koskaan viedä tietokantaan. Mikäli haettavaa vaihetta tai tolppaa ei löydy muistiratkaisusta, ajetaan päivitysfunktio, joka hakee tietokannasta uudet tiedot. Tämän jälkeen hakua yritetään uudelleen.

Nopea muistiratkaisu mahdollistaa myös algoritmin tehokkaan toiminnan, joten sen rooli on tärkeä algoritmin toiminnan kannalta, varsinkin jos algoritmin piirissä olevien tolppien lukumäärä tulee nousemaan vielä nykyisestä määrästä.

4.5 Muuttujat

Algoritmin toteutuksessa käytetään useita muuttujia, jotka ovat oleellisia ohjelman toiminnan kannalta. Osa näistä muuttujista on jo esitelty osittain kolmannessa luvussa samalla kun esiteltiin algoritmin toimintaa, mutta niiden toiminta saattaa hieman poiketa skenaarioiden ja ongelman esittelyn yhteydessä esitetystä tavasta. Eroavaisuudet johtuvat siitä, että skenaarioiden yhteydessä asioita on yksinkertaistettu hieman, mutta todellisten muuttujien tulee toimia geneerisesti kaikille eri vaiheille ja niiden sulakkeille. Tällöin arvoja ei voida asettaa suoraan tiettyyn tasoon, vaan muuttujia käytetään apuna tasojen laskemiseen. Tästä esimerkkinä seuraavaksi esiteltävät ylä- ja alapuskuri-muuttujat, joiden avulla lasketaan ylä- ja alarajat.

Yläpuskuri on muuttuja, jonka avulla lasketaan yläraja, eli raja, jonka jälkeen virrankulutusta ruvetaan laskemaan. Käytännössä yläraja saadaan vähentämällä sulakkeesta yläpuskuri-muuttuja, esimerkiksi jos sulake on 16 ampeeria ja yläpuskuri 2 ampeeria, saadaan ylärajaksi 14 ampeeria. Ylärajaa ei ole suoraan mahdollista asettaa, sillä vaiheiden sulakkeiden koot saattavat vaihdella. Yläpuskurin tilasta suoraan asetettu yläraja, esimerkiksi 14 ampeeria, olisi hyvä yläraja sulakkeen ollessa 16 ampeeria, mutta aivan liian matala sulakkeen ollessa 32 ampeeria. Tästä syystä käytetään yläpuskuria, jolloin 2 ampeerin yläpuskurilla 16 ampeerin sulakkeen yläraja on 14 ampeeria ja 32 ampeerin sulakkeen yläraja on 30 ampeeria.

Alapuskuri on muuttuja, jonka avulla lasketaan alaraja, eli raja, jonka jälkeen virrankulutusta nostetaan. Jos alapuskuri on 4 ampeeria ja sulake 16 ampeeria, niin alarajaksi saadaan 12 ampeeria.

Maksimiaika on muuttuja, jonka perusteella lasketaan aktiiviset tolpat. Algoritmi arvioi tolppien tilannetta vain virrankulutusviestien pohjalta, joka tarkoittaa sitä, että vain sellaiset tolpat huomioidaan, joiden virrankulutus on suurempi kuin 0. Vaiheen virrankulutuksen laskemisessa on tärkeää huomioida vain sellaiset tolpat, jotka ovat aktiivisia, sillä muuten tulos saattaisi vääristyä. Tilannetta voisi havainnoida seuraavanlaisesti: tolppa kuluttaa 10 ampeeria, jonka jälkeen tolppa otetaan pois verkosta ja ei enää lähetä virrankulutusviestejä. Algoritmin muistiin viimeiseksi arvoksi jää 10 ampeeria, joka tarkoittaa, että vaiheen virrankulutuslaskuissa on 10 ampeeria ylimääräistä kulutusta, josta seuraisi, että algoritmi tekee vääriä päätöksiä. Tämän vuoksi käytetään maksimiaika-muuttujaa, joka määrittää minuuteissa, kuinka monta minuuttia tolpan edellisestä virrankulutusviestistä saa olla kulunut maksimissaan minuutteja, jotta tolppa voidaan ottaa laskuihin mukaan. Maksimiarvoa verrataan tolpalla olevaan päivitysaika-muuttujaan.

Minimilaskumäärä on muuttuja, joka määrittää kuinka paljon kulutusta vähintään lasketaan ylärajan ylittyessä. Esimerkiksi jos sulake on 16 ampeeria, yläraja 15 ampeeria, minimilaskumäärä 4 ampeeria, niin virrankulutus lasketaan aina vähintään 12 ampeeriin, vaikka virrankulutus olisi 25 ampeeria tai 16 ampeeria. Eli minimilaskumäärä määrittää

minimivirrankulutustavoitteen laskemalla sulakkeen arvosta, kuinka paljon kulutusta lasketaan. Minimilaskumäärä-muuttujaa asettaessa on hyvä huomioida, ettei sen kannata laskea kulutusta alarajan alapuolelle, koska tästä seuraisi jälleen kulutuksen nostaminen algoritmin toimesta. Alarajan ja minimipudotuksen välissä olisi hyvä siis olla pieni turvaväli, jotta vältytään sahaamiselta.

Maksiminostomäärä on muuttuja, joka määrittää kuinka paljon virrankulutusta voidaan kerralla nostaa poistamalla EVSE-tolppien rajoituksia, mikäli kulutus alittaa alarajamuuttujan. Arvo kannattaa todennäköisesti pitää pienenä, jottei kulutusta nosteta liian nopeasti yhdellä kerralla. Kulutusta ei kuitenkaan nosteta alarajan yläpuolelle. Jos esimerkiksi alaraja on 10 ampeeria, maksiminostomäärä on 2 ampeeria, tolppia on rajoitettu 5 ampeeria ja vaiheen kokonaisvirrankulutus on 7 ampeeria. Tällöin ensimmäisellä kerralla nostetaan virrankulutusta rajoituksia poistamalla 2 ampeerin verran, jolloin virrankulutus nousee 9 ampeeriin ja rajoitusten määrä laskee viidestä kolmeen ampeeriin. Toisella kerralla virrankulutusta nostetaankin vain 1 ampeeri, koska alarajaa ei ylitetä. Virrankulutus nousee 10 ampeeriin ja rajoitusten määrä laskee kahteen ampeeriin.

Minimikäynnistystila on muuttuja, joka määrittää kuinka paljon vapaata tilaa vaiheen virrankulutuksessa tulee olla, jotta sammutettuja EVSE-tolppia voidaan käynnistää uudelleen. Koska EVSE-tolppien minimivirrankulutus on 6 ampeeria, minimikäynnistystila-muuttujan on pakko olla suurempi kuin 6 ampeeria, jotta latauslaite voidaan ylipäätänsä käynnistää. Minimikäynnistystila-muuttuja lasketaan alarajasta, eli alarajan alapuolella tulee olla minimikäynnistystila-muuttujan arvon verran tilaa, jotta tolppa voidaan käynnistää.

Näiden muuttujien avulla algoritmi tekee päätöksiä ja laskee mitä toimenpiteitä sen tulee tehdä. Seuraavaksi esitellään tarkemmin algoritmin toteutusta.

4.6 Algoritmin ja vuokaavion läpikäynti

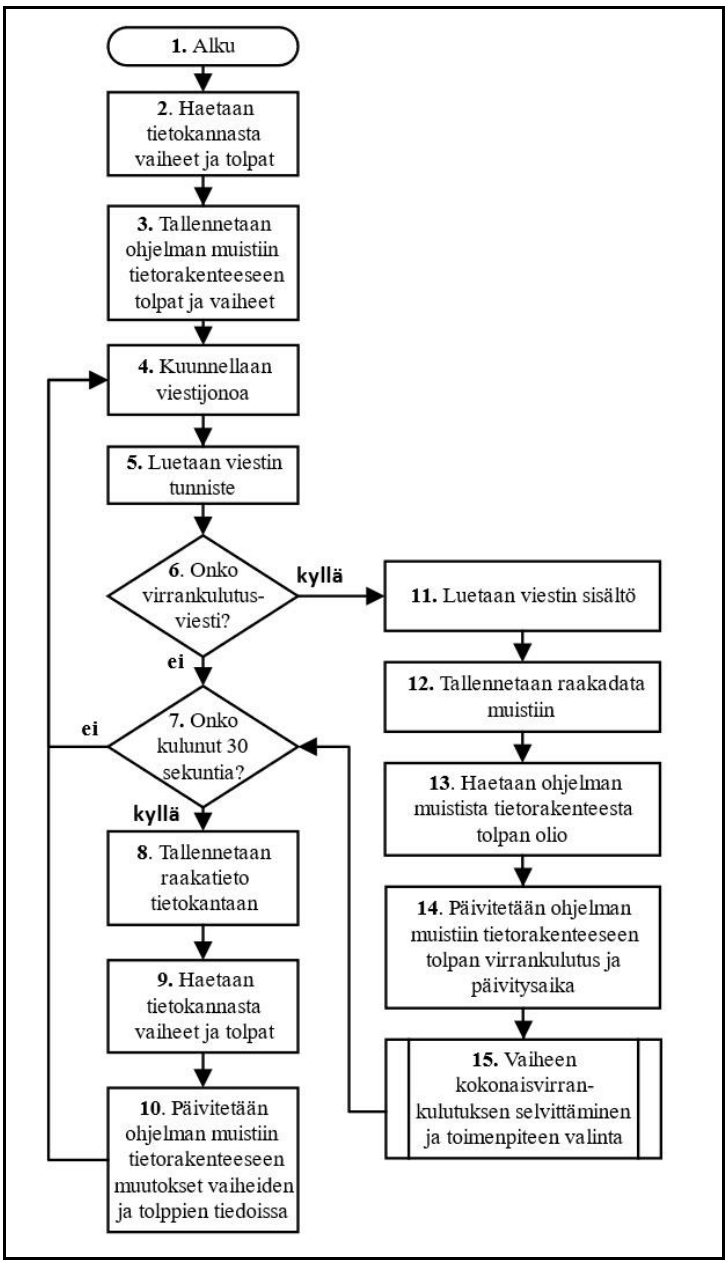
Seuraavaksi käydään läpi algoritmin toteutusta vuokaavioiden avulla. Toteutus on jaettu neljään osaan, joista ensimmäiseksi käydään läpi algoritmin perustoiminnot ja viestien lukeminen.

4.6.1 Perustoiminnot ja viestien lukeminen

Kuvassa 28 on kuvattu ensimmäisen osan tärkeimmät vaiheet. Algoritmin käynnistyessä haetaan tietokannasta vaiheet ja tolpat näille vaiheille ja tallennetaan ne muistiin tietorakenteeseen (kohdat 2 ja 3), kuten aikaisemmin kuvattiin tietokanta- ja muistiratkaisun esittelyjen yhteydessä.

Tarvittavien alustusten jälkeen algoritmi alkaa kuuntelemaan viestijonoa (kohta 4). Kun algoritmi saa viestin, tarkistetaan, onko se tolpilta tuleva virrankulutusviesti (kohta 5 ja 6). Jos on, niin luetaan viestin sisältö ja tallennetaan viestin sisältämä tieto raakatietona ohjelman muistiin odottamaan tietokantaan lisäystä (kohdat 11 ja 12). Tämän jälkeen

tietorakenteesta haetaan viestissä olevan tunnisteen perusteella ohjelman muistista tietorakenteesta kyseinen tolppaolio (kohta 13). Tälle oliolle päivitetään uusi virrankulutustieto, joka viestissä saapui, ja olion päivitysaika päivitetään vastaamaan sen hetkistä ajankohtaa (kohta 14). Tämän jälkeen siirrytään kuormantasausalgoritmin ensimmäiseen funktioon, jota käsitellään seuraavassa kohdassa (kohta 15).



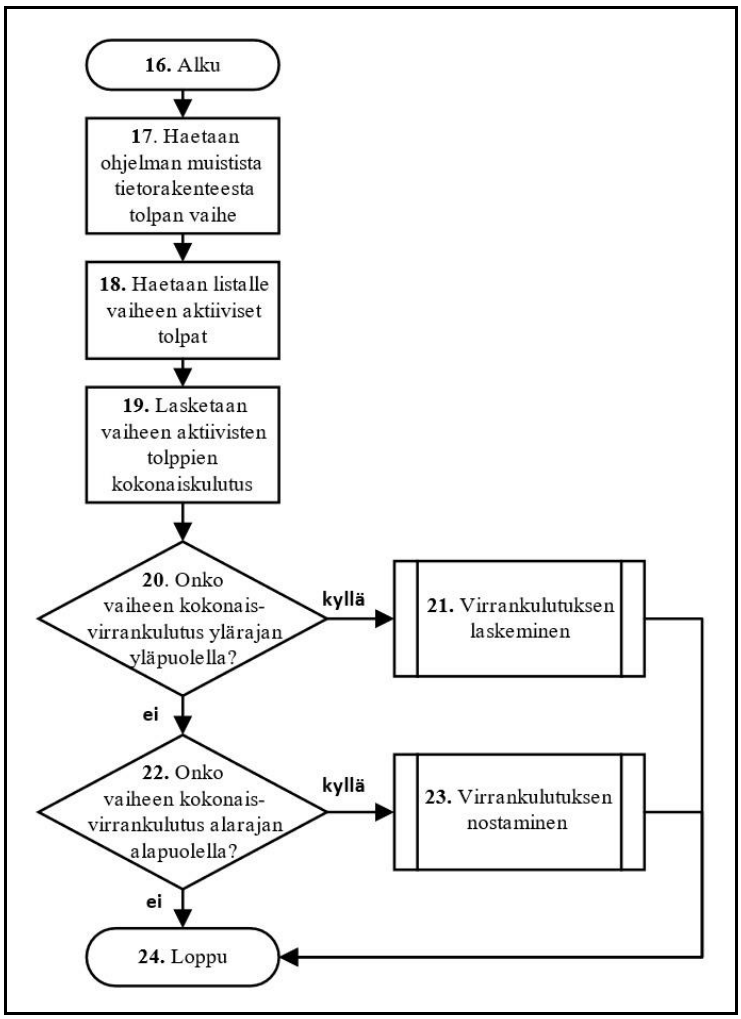
Kuva 28. Vuokaavio viestien lukemisesta.

Kun kohdasta 15 palataan takaisin tai jos saapunut viesti ei ollut virrankulutusviesti, niin tarkistetaan, kuinka monta sekuntia on kulunut edellisestä raakatiedon viennistä tietokantaan (kohta 7). Jos tästä on kulunut yli 30 sekuntia, raakatieto viedään tietokantaan (kohta 8). Samassa yhteydessä myös nollataan sekuntilaskuri. Tämän jälkeen päivitetään myös tietokannasta ohjelman muistissa olevaan tietorakenteeseen vaiheiden ja tolppien

tiedot (kohdat 9 ja 10). Tämän jälkeen palataan kuuntelemaan taas viestijonoa. Mikäli ei ole kulunut yli 30 sekuntia, palataan suoraan kuuntelemaan viestijonoa.

4.6.2 Vaiheen kokonaisvirrankulutuksen selvittäminen ja toimenpiteen valinta

Toisessa osassa selvitetään vaiheen aktiivisten tolppien kokonaisvirrankulutus ja arvioidaan, tarvitseeko virrankulutukselle tehdä toimenpiteitä (kuva 29). Aluksi ohjelman muistissa olevasta tietorakenteesta haetaan tolpalle vaiheolio (kohta 17). Tämän jälkeen haetaan listalle aktiiviset tolpat (kohta 18). Tämä tapahtuu vertaamalla tolppaolion päivitysaikaa kohdassa 4.5 aikaisemmin esiteltyyn maksimiaika-muuttujaan. Mikäli algoritmi on saanut virrankulutusviestin tolpalta maksimiajan (esimerkiksi 5 minuutin) sisällä, tolppa lisätään listalle. Seuraavaksi lasketaan tämän aktiivisten tolppien listan kokonaisvirrankulutus (kohta 19).



Kuva 29. Vuokaavio vaiheen kokonaisvirrankulutuksen selvittämisestä ja tarvittavan toimenpiteen valitsemisesta.

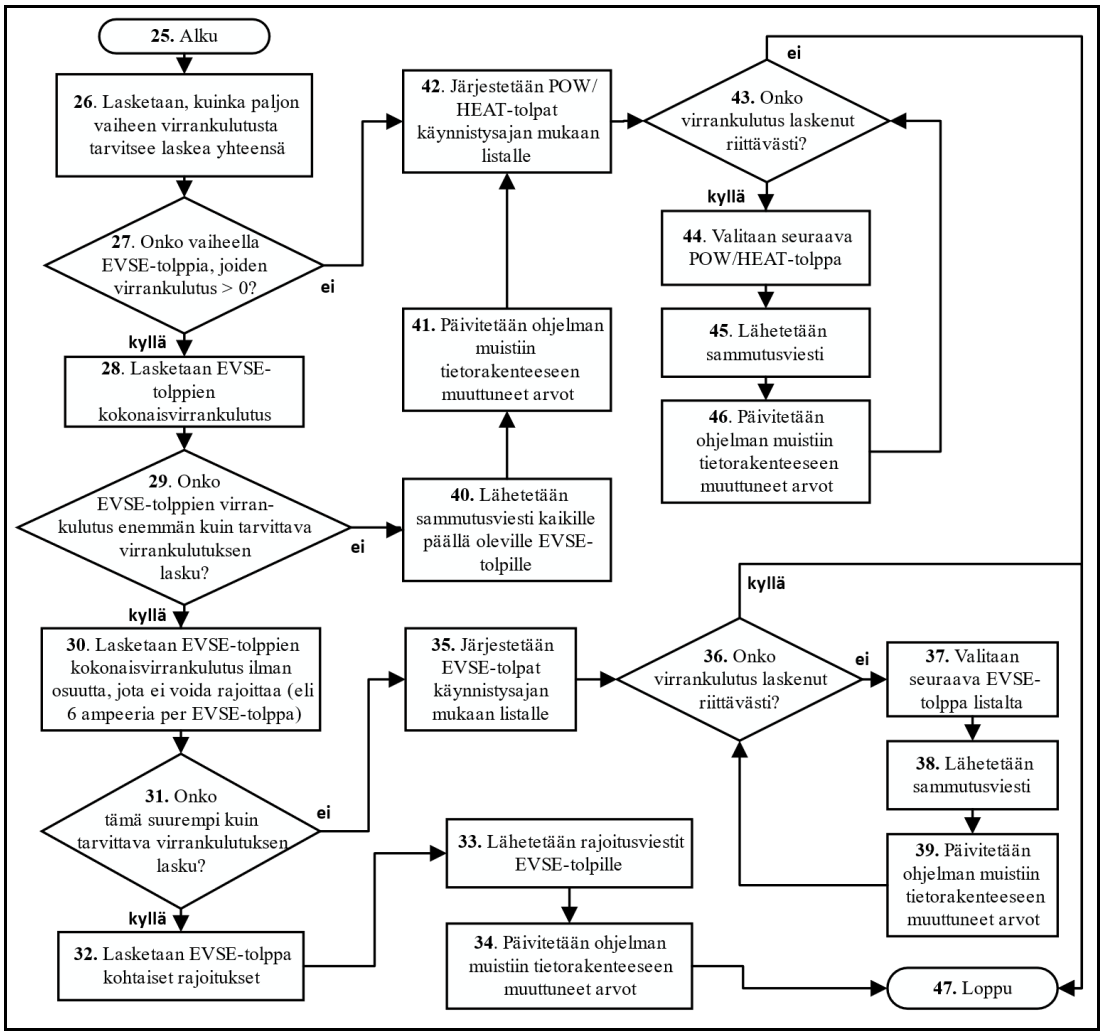
Ensiksi tarkistetaan ylittääkö kokonaisvirrankulutus ylärajan (kohta 20). Tässä tarkistuksessa käytetään avuksi yläpuskuri-muuttujaa. Mikäli virrankulutus ylittää ylärajan,

siirrytään virrankulutuksen laskemiseen, jota käsitellään alakohdassa 4.6.3. Mikäli virrankulutus ei ylitä ylärajaa, tarkistetaan alittaako se alarajan (kohta 21). Tässä tarkistuksessa käytetään avuksi alapuskuri-muuttujaa. Mikäli virrankulutus alittaa alarajan, siirrytään virrankulutuksen nostamiseen, jota käsitellään alakohdassa 4.6.4. Mikäli virrankulutus ei ole alarajankaan alapuolella, siirrytään pois funktiosta takaisin alakohdan 4.6.1 kohtaan 7.

4.6.3 Virrankulutuksen laskeminen

Virrankulutuksen laskemiseen pätee luvussa 3 esitetyt järjestykset, joissa tolppia rajoitetaan ja sammutetaan. Ensisijaisesti tehdään toimenpiteitä EVSE-tolpille ja sitten vasta POW/HEAT-tolpille.

Kuvassa 30 on kuvattu vuokaaviona funktio virrankulutuksen laskemiseen. Aluksi lasketaan, kuinka paljon vaiheen virrankulutusta tarvitsee laskea yhteensä (kohta 26). Tässä laskussa käytetään avuksi minimilaskumäärä-muuttujaa, jonka avulla lasketaan, kuinka paljon vaiheen kokonaisvirrankulutusta täytyy yhteensä madaltaa. Tämän jälkeen tarkistetaan, onko vaiheella yli 0 ampeeria kuluttavia EVSE-tolppia (kohta 27).



Kuva 30. Vuokaavio vaiheen virrankulutuksen laskemisesta.

Jos ei ole yli 0 ampeeria kuluttavia EVSE-tolppia, siirrytään sammuttamaan POW/HEAT-tolppia. Aluksi POW/HEAT-tolpat järjestetään käynnistysajan mukaan (kohta 42). Tämän jälkeen sammutetaan tolppia, kunnes virrankulutus on laskenut tarpeeksi (kohdat 43, 44, 45 ja 46). Jokaisen sammutusviestin jälkeen ohjelman muistiin tietorakenteeseen päivitetään muuttuneet arvot. Kun tarpeeksi POW/HEAT-tolppia on sammutettu, siirrytään pois funktiosta.

Jos yli 0 ampeeria kuluttavia EVSE-tolppia löytyy, lasketaan EVSE-tolppien kokonaisvirrankulutus (kohta 28). Tämän jälkeen tarkistetaan riittääkö vain EVSE-tolppien rajoittaminen vai pitääkö myös POW/HEAT-tolppia sammuttaa (kohta 29). Mikäli EVSE-tolppien virrankulutus on pienempi kuin tarvittava kokonaisvirrankulutuksen lasku, niin sammutetaan kaikki päällä olevat EVSE-tolpat (kohta 40) ja päivitetään ohjelman muistiin tietorakenteeseen muuttuneet arvot (kohta 41), jonka jälkeen siirrytään kohtaan 42 eli sammuttamaan POW/HEAT-tolppia.

Mikäli EVSE-tolppien virrankulutus on suurempi kuin tarvittava kokonaisvirrankulutuksen lasku, niin selvitään vain EVSE-tolppia koskevilla toimenpiteillä. Kuten aikaisemmin on esitelty laitteiden yhteydessä, EVSE-tolppa voidaan rajoittaa minimissään 6 ampeeriin, jonka jälkeen se joudutaan sammuttamaan, mikäli virrankulutusta tarvitsee vielä laskea. Seuraavaksi tarkistetaan, riittääkö vain EVSE-tolppien rajoittaminen vai tarvitseeko niitä myös sammuttaa. Tämä tehdään laskemalla EVSE-tolppien virrankulutuksesta osuus, jota on mahdollista rajoittaa (kohta 30), eli vähentämällä kulutuksesta 6 ampeeria kertaa tolppien lukumäärä. Seuraavaksi tarkistetaan, riittääkö vain EVSE-tolppien rajoittaminen, eli verrataan sitä tarvittavaan kokonaisvirrankulutuksen laskuun, joka aluksi laskettiin (kohta 31).

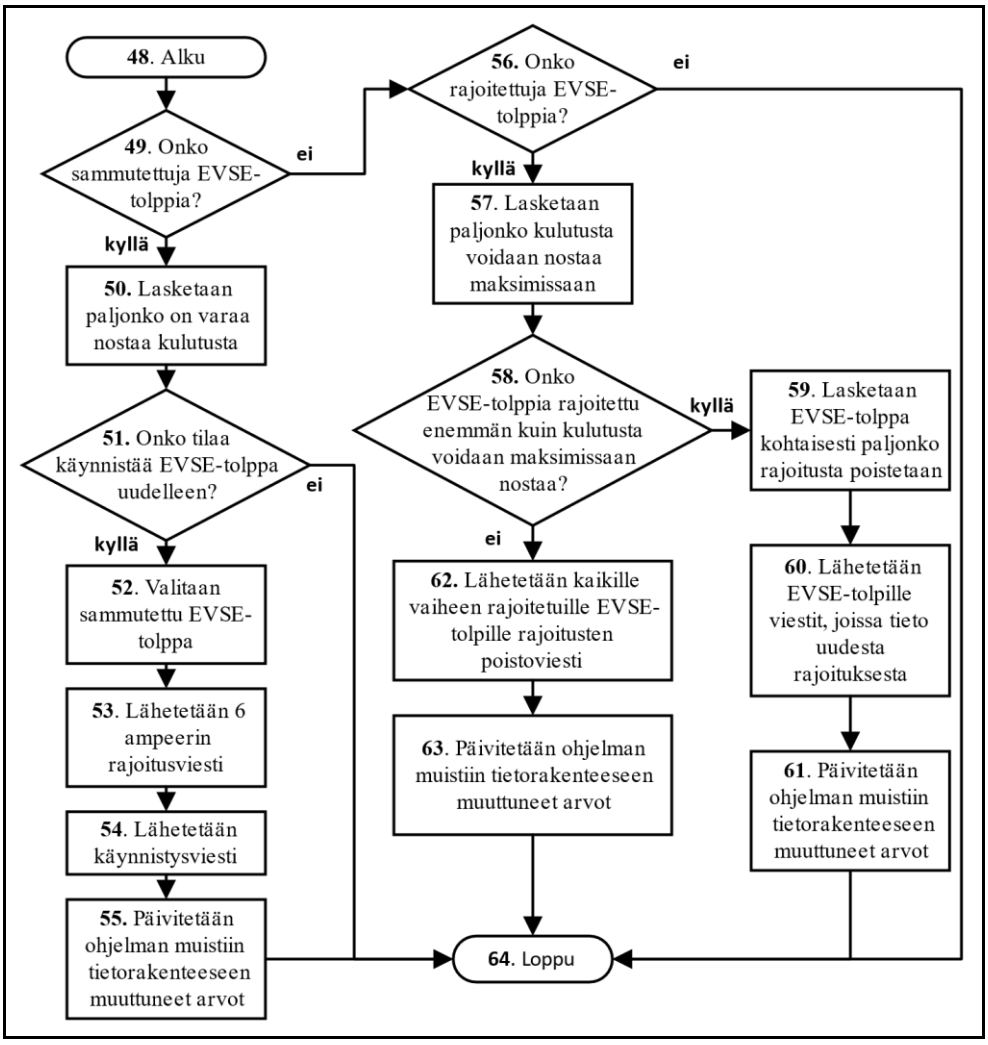
Jos EVSE-tolppia on mahdollista rajoittaa tarpeeksi, lasketaan EVSE-tolppa kohtaiset rajoitukset (kohta 32), jonka jälkeen lähetetään tolpille viestit uusista virrankulutuksista (kohta 33) ja lopuksi päivitetään ohjelman muistin tietorakenteeseen muuttuneet arvot.

Jos EVSE-tolppia ei ole mahdollista rajoittaa tarpeeksi, järjestetään EVSE-tolpat käynnistysajan mukaan (kohta 35). Tämän jälkeen EVSE-tolppia sammutetaan, kunnes virrankulutus on laskenut tarpeeksi (kohdat 36, 37, 38 ja 39). Tolpille lähetetään sammutusviesti ja muuttuneet arvot päivitetään ohjelman tietorakenteeseen. Samalla merkataan sammutettavalle tolppaoliolle tieto siitä, että se on sammutettu, eli tolpan onkoSammutettu-muuttuja asetetaan ”True”-tilaan (kuva 27).

4.6.4 Virrankulutuksen nostaminen

Virrankulutuksen nostamiseen pätevät samat ehdot, jotka esiteltiin luvussa 3. Virrankulutusta voidaan nostaa joko käynnistämällä EVSE-tolppia tai poistamalla niiden rajoituksia. Kuvassa 31 on esitetty vuokaaviossa algoritmin toiminta virrankulutuksen nostamiseen.

Aluksi tarkistetaan, onko sammutettuja EVSE-tolppia (kohta 49). Jos on, niin siirrytään laskemaan, paljonko kulusta voidaan nostaa (kohta 50), jonka jälkeen katsotaan, onko vaiheen virrankulutuksessa tarpeeksi tilaa, jotta EVSE-tolppa voitaisiin käynnistää uudelleen (kohta 51). Tässä käytetään apuna minimikäynnistystila-muuttujaa, joka määrittää kuinka paljon tilaa tulee olla, jotta tolppa voidaan käynnistää uudelleen. Jos vaiheen virrankulutuksessa ei ole tilaa tarpeeksi tolpan käynnistämiseen, siirrytään pois funktiosta.



Kuva 31. Vuokaavio vaiheen virrankulutuksen nostamisesta.

Jos tilaa on tarpeeksi, valitaan ensimmäinen sammutettu EVSE-tolppa (kohta 52), jonka jälkeen sille lähetetään viesti, joka asettaa sen virrankulutuksen 6 ampeeriin (kohta 53). Tämän jälkeen tolppa käynnistetään lähettämällä käynnistysviesti (kohta 54). Uudet arvot päivitetään ohjelman tietorakenteeseen (kohta 55).

Mikäli vaiheella ei ole sammutettuja EVSE-tolppia, tarkistetaan, onko sillä rajoitettuja EVSE-tolppia (kohta 56). Jos ei, niin siirrytään pois funktiosta. Jos on, niin lasketaan,

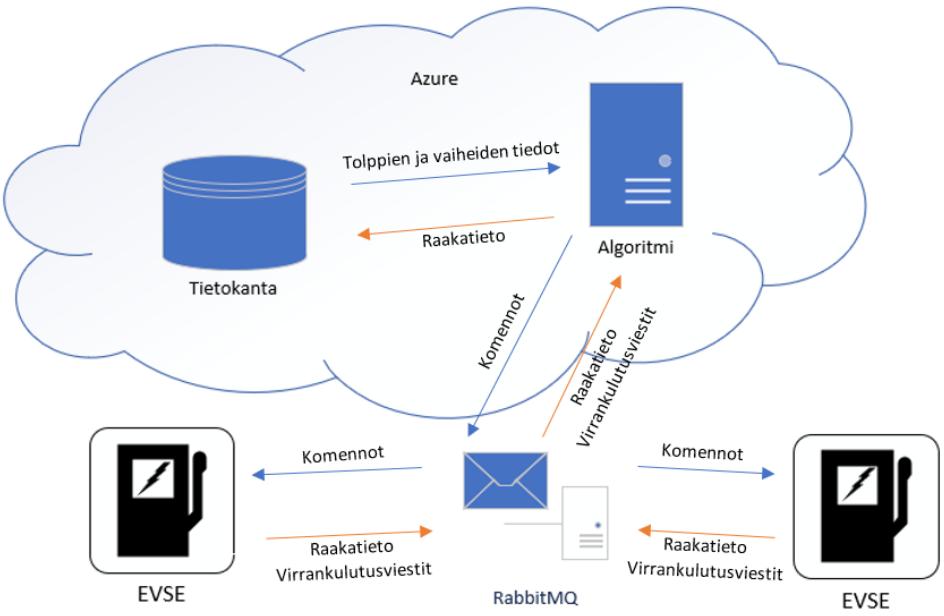
kuinka paljon kulutusta voidaan maksimissaan nostaa. Apuna tässä käytetään maksiminostomäärä-muuttujaa, joka määrittää kuinka paljon yhdellä kerralla maksimissaan virrankulutusta voidaan nostaa (kohta 57).

Jos EVSE-tolppien yhteenlasketut rajoitukset ovat enemmän kuin maksiminostomäärä (kohta 58), lasketaan tolppakohtaisesti, paljonko rajoituksia poistetaan (kohta 59). Tämän jälkeen EVSE-tolpille lähetetään viestinä komento muuttaa virrankulutus uuteen laskettuun arvoon (kohta 60). Nämä arvot päivitetään myös ohjelman tietorakenteeseen tolppaolioille (kohta 61). Tämän jälkeen funktiosta poistutaan.

Jos EVSE-tolppien yhteenlasketut rajoitukset ovat vähemmän kuin maksiminostomäärä (kohta 58), voidaan kaikki rajoitukset poistaa suorilta. Tämä tehdään lähettämällä kaikille rajoitetuille EVSE-tolpille viestinä komento nostaa virrankulutus takaisin maksimilatausvirtaan (kohta 62). Viestien lähettämisen jälkeen ohjelman tietorakenteeseen olioille päivitetään uudet muuttuneet arvot. Tämän jälkeen funktiosta poistutaan.

4.7 Arkkitehtuuri

Kuvassa 32 on kuvattu koko ratkaisun arkkitehtuurimalli. Algoritmi ja tietokanta toimivat Azure-pilvipalvelussa. Algoritmi tallentaa tietokantaan raakatietoa ja hakee tietokannasta tolppien ja vaiheiden tietoja 30 sekunnin välein. Algoritmi saa RabbitMQ:n kautta viestejä tolpilta ja tekee tarvittavia ratkaisuja niiden pohjalta. Mahdolliset tarvittavat komennot lähetetään viesteinä RabbitMQ:n kautta tolpille.



Kuva 32. Koko ratkaisun arkkitehtuurimalli.

5 Kuormantasausalgoritmin testaus ja tulokset

Tässä luvussa kuvataan algoritmin oikeellisuuden testaus. Algoritmille asetetaan kuusi vaatimusta, jotka sen tulee täyttää. Nämä vaatimukset ja testit, joilla vaatimuksia testataan, esitellään kohdassa 5.1. Kohdassa 5.2 käydään läpi testien tulokset ja kohdassa 5.3 on yhteenveto tuloksista ja havaintoja testauksesta ja testeistä.

Algoritmin testaus toteutetaan testidatalla luomalla olioita ohjelman tietorakenteeseen ja tämän jälkeen ajamalla algoritmi. Kaikki testit ajetaan samalla tietokoneella. Testidatalla on mahdollista luoda juuri halutunlaisia tilanteita, joita on vaikeampi toistaa oikeassa ympäristössä.

5.1 Vaatimusmäärittely

Seuraavaksi esitellään vaatimukset, jotka algoritmin tulee täyttää toimiakseen oikeellisesti. Vaatimuksia on kuusi, joista jokainen on erilainen tilanne, jossa algoritmin tulee toimia eri tavoin laskeakseen tai nostaakseen vaiheen kokonaisvirrankulutusta. Jokaisen vaatimuksen testaamiseen on suunniteltu vähintään kaksi testiä. Seuraavaksi käydään läpi vaatimukset ja testit.

5.1.1 Vaatimus 1: Algoritmi osaa laskea kulutusta rajoittamalla EVSE-tolppia

Ensimmäinen vaatimus algoritmille on, että algoritmi osaa laskea virrankulutusta rajoittamalla EVSE-tolppia. Kuten aikaisemmin ongelman ja algoritmin esittelyn yhteydessä on kerrottu, algoritmi voi rajoittaa EVSE-tolppia 6 ampeeriin asti. Taulukossa 1 on esitetty vaatimuksen testaamiseen tarkoitetut testit, joita on kolme kappaletta. Testeissä muuttujina käytetään yläpuskuri-muuttujaa, joka määrittää ylärajan sulakkeesta, sekä minimilasku-muuttujaa, joka määrittää vähimmäismäärän sille, paljonko virrankulutusta lasketaan sulakkeesta.

Ensimmäinen testeistä testaa yksittäisen latauslaitteen rajoittamista. Toinen testeistä testaa kahden latauslaitteen rajoittamista, kun toinen latauslaitteista on jo rajoitettu minimiin. Kolmas testi testaa kolmen latauslaitteen rajoittamista, kun latauslaitteet kuluttavat kaikki eri määrän virtaa.

Taulukko 1. Testit ensimmäisen vaatimuksen testaamiseen.

Nro.	Muuttujat	Kokoonpano	Testitapaus	Odotettu tulos
1.1	Yläpuskuri: 1 Minimilasku: 4	Sulake: 16A 1. EVSE: 16A	Yksittäisen latauslaitteen rajoittaminen.	EVSE-tolppaa rajoitetaan 16 ampeerista 12 ampeeriin.
1.2	Yläpuskuri: 1 Minimilasku: 4	Sulake: 16A 1. EVSE: 10A 2. EVSE: 6A	Vaiheen rajoittaminen, kun on kaksi latauslaitetta, joista toinen on jo minimissä.	Ensimmäinen EVSE-tolppa rajoitetaan 10 ampeerista 6 ampeeriin. Kokonaiskulutus laskee 16 ampeerista 12 ampeeriin.
1.3	Yläpuskuri: 2 Minimilasku: 6	Sulake: 32A 1. EVSE: 16A 2. EVSE: 10A 3. EVSE: 8A	Kolmen latauslaitteen rajoittaminen.	Kokonaisvirrankulutus laskee 34 ampeerista 26 ampeeriin. Ensimmäistä EVSE-tolppaa rajoitetaan 5 ampeeria, toista 2 ampeeria, kolmatta 1 ampeeri.

5.1.2 Vaatimus 2: Algoritmin osaa laskea kulutusta sammuttamalla EVSE-tolppia

Toinen vaatimus on, että algoritmi osaa laskea vaiheen kokonaisvirrankulutusta sammuttamalla EVSE-tolppia. Jos ei ole enää mahdollista rajoittaa latauslaitetta, seuraava vaihtoehto on EVSE-tolpan sammuttaminen. Taulukossa 2 on esitetty testit vaatimukselle 2. Testeissä käytetään yläpuskuri-muuttujaa, joka määrittää ylärajan sulakkeesta, sekä minimilasku-muuttujaa, joka määrittää kuinka paljon kulutusta lasketaan.

Ensimmäisessä testissä on kolme latauslaitetta, jotka on kaikki rajoitettu minimiin. Testin tarkoitus on selvittää, osaako algoritmi sammuttaa pisimpään päällä olleen latauslaitteen. Toisessa testissä on kaksi latauslaitetta ja testataan sitä, osaako algoritmi sammuttaa toisen ja rajata toista EVSE-tolppaa. Minimilasku-muuttuja on asetettu epärealistisen korkeaksi, jotta testi voidaan toteuttaa. Yleensä yhden EVSE-tolpan sammuttaminen riittää, sillä se yksinään laskee kokonaisvirrankulutusta jo 6 ampeeria.

Taulukko 2. Testit toisen vaatimuksen testaamiseen.

Nro.	Muuttujat	Kokoonpano	Testitapaus	Odotettu tulos
2.1	Yläpuskuri: 1 Minimilasku: 2	Sulake: 16A 1. EVSE: 6A 2. EVSE: 6A 3. EVSE: 6A	Kolme latauslaitetta, jotka kaikki jo rajoitettu minimiin eli 6 ampeeriin.	Pisimpään päällä ollut latauslaite latauslaitteista sammutetaan. Kokonaisvirrankulutus laskee 18 ampeerista 12 ampeeriin.
2.2	Yläpuskuri: 2 Minimilasku: 24	Sulake: 32A 1. EVSE: 16A 2. EVSE 15A	Kaksi latauslaitetta, jotka käyttävät 16 ampeeria ja 15 ampeeria virtaa.	Kokonaisvirrankulutus laskee 31 ampeerista 8 ampeeriin. Toinen latauslaite sammutetaan ja toinen rajoitetaan 8 ampeeriin.

5.1.3 Vaatimus 3: Algoritmi osaa laskea kulutusta sammuttamalla POW ja HEAT-tolppia

Kolmas vaatimus on, että algoritmi osaa laskea vaiheen kokonaisvirrankulutusta sammuttamalla POW ja HEAT-tolppia. Sammutus tehdään sammuttamalla tolppia päälläoloajan mukaan. Taulukossa 3 on esitetty testit, joita käytetään vaatimuksen testaamiseen. Testeissä käytetään yläpuskuri-muuttujaa, joka määrittää ylärajan sulakkeesta ja minimilasku-muuttujaa, joka määrittää kuinka paljon kulutusta tulee vähintään laskea.

Taulukko 3. Testit kolmannen vaatimuksen testaamiseen.

Nro.	Muuttujat	Kokoonpano	Testitapaus	Odotettu tulos
3.1	Yläpuskuri: 1 Minimilasku: 3	Sulake: 16A 1. POW: 5A 2. HEAT: 5A 3. HEAT: 5A	Kolme POW ja HEAT-tolppaa, jotka kaikki kuluttavat 5 ampeeria.	Pisimpään päällä ollut tolppa sammutetaan.
3.2	Yläpuskuri: 1 Minimilasku: 12	Sulake: 16A 1. HEAT: 4A 2. HEAT 4A 3. HEAT: 4A 4. POW 4A	Neljä POW- ja HEAT-tolppaa, jotka kaikki kuluttavat 4 ampeeria.	Kolme tolpista sammutetaan käynnistysjärjestyksen mukaan.

Ensimmäisessä testissä on kolme POW ja HEAT-tolppaa, joista pisimpään päällä olleen tulisi sammua. Toisessa testissä on 4 POW ja HEAT-tolppaa, joista kolmen tulisi sammua käynnistysjärjestyksen mukaisesti.

5.1.4 Vaatimus 4: Algoritmi osaa laskea kulutusta sammuttamalla EVSE-, POW- ja HEAT-tolppia

Vaatimus neljä testaa osaako algoritmi laskea kulutusta sammuttamalla sekä EVSE-, POW- että HEAT-tolppia. Ensimmäisenä sammutetaan EVSE-tolpat, jonka jälkeen sammutetaan tarvittava määrä POW- ja HEAT-tolppia. Taulukossa 4 on kuvattu testit vaatimuksen testaamiseen. Muuttujina testeissä on yläpuskuri, joka määrittää sulakkeesta yläarajan, sekä minimilasku, joka määrittää kuinka paljon kulutusta pudotetaan sulakkeesta.

Ensimmäisessä testissä on yksi EVSE-tolppa ja kaksi HEAT-tolppaa. Algoritmin tulisi sammuttaa EVSE-tolppa sekä pisimpään päällä ollut HEAT-tolppa. Toisessa testissä on kaksi EVSE-tolppaa ja kolme POW- ja HEAT-tolppaa. Oikein toimiessaan algoritmi sammuttaa molemmat EVSE-tolpat ja kaksi POW- ja HEAT-tolpista päälläoloajan mukaan.

Taulukko 4. Testit neljännen vaatimuksen testaamiseen.

Nro.	Muuttujat	Kokoonpano	Testitapaus	Odotettu tulos
4.1	Yläpuskuri: 1 Minimilasku: 10	Sulake: 16A 1. EVSE: 6A 2. HEAT: 5A 3. HEAT: 5A	Yksi EVSE-tolppa ja kaksi HEAT-tolppaa, jotka yhteensä kuluttavat 16 ampeeria.	EVSE-tolppa ja pisimpään päällä ollut HEAT-tolppa sammutetaan.
4.2	Yläpuskuri: 2 Minimilasku: 23	Sulake: 32A 1. EVSE: 9A 2. EVSE: 8A 3. POW: 5A 4. HEAT: 5A 5. POW: 5A	Useita erilaisia tolppia, jotka yhteensä kuluttavat virtaa 32 ampeeria.	Molemmat EVSE-tolpat sammutetaan. Kaksi POW- ja HEAT-tolppaa sammutetaan käynnistysaikojen mukaan.

5.1.5 Vaatimus 5: Algoritmi osaa nostaa kulutusta käynnistämällä EVSE-tolppia

Viides vaatimus on, että algoritmi osaa nostaa vaiheen kokonaisvirrankulutusta käynnistämällä aikaisemmin sammutettuja EVSE-tolppia uudelleen. Taulukossa 5 on kuvattu testit vaatimuksen testaamiseen. Muuttujina käytetään alapuskuri-muuttujaa, joka määrittää alarajan sulakkeesta, sekä minimikäynnistystila-muuttujaa, joka määrittää kuinka paljon

alarajan ja virrankulutuksen välillä täytyy olla tilaa, jotta latauslaite voidaan käynnistää uudelleen.

Ensimmäisessä testissä on kaksi EVSE-tolppaa, joista toinen on sammutettu. Algoritmin pitää käynnistää se uudelleen oikein toimiessaan. Toisessa testissä on kaksi EVSE-tolppaa, joista toinen on sammutettu. Algoritmin ei pitäisi oikein toimiessaan käynnistää tolppaa uudelleen, sillä alarajan ja virrankulutuksen välissä ei ole minimikäynnistystilan verran tilaa.

Taulukko 5. Testit viidennen vaatimuksen testaamiseen.

Nro.	Muuttujat	Kokoonpano	Testitapaus	Odotettu tulos
5.1	Alapuskuri: 4 Minimikäynnistystila: 7	Sulake: 32A 1. EVSE: 0A 2. EVSE: 6A	Kaksi EVSE-tolppaa, joista toinen on sammutettu.	Sammutettu EVSE-tolppa käynnistetään uudelleen ja sen virrankulutus nousee 6 ampeeriin.
5.2	Alapuskuri: 2 Minimikäynnistystila: 7	Sulake 16A 1. EVSE: 0A 2. EVSE: 8A (rajoitettu 8A)	Kaksi EVSE-tolppaa, joista ensimmäinen on sammutettu ja toinen rajoitettu.	Tolppaa ei käynnistetä uudelleen, koska alarajan alapuolella ei ole tarpeeksi tilaa. Rajoituksia ei poisteta, koska vaiheella on sammutettu tolppa.

5.1.6 Vaatimus 6: Algoritmi osaa nostaa kulutusta poistamalla EVSE-tolppien rajoituksia

Kuudes ja viimeinen vaatimus on, että algoritmi osaa nostaa vaiheen kokonaisvirrankulutusta poistamalla EVSE-tolppien rajoituksia. Taulukossa 6 on esitetty testit vaatimuksen testaamista varten. Muuttujina toimii alapuskuri, joka määrittää alarajan sulakkeesta, sekä maksiminosto-muuttuja, joka määrittää kuinka paljon rajoituksia voidaan yhdellä kerralla poistaa vaiheen latauslaitteilta.

Ensimmäisessä testissä on kaksi rajoitettua EVSE-tolppaa. Oikein toimiessaan algoritmi poistaa rajoituksia molemmilta tolpilta maksimissaan maksiminosto-muuttujan verran. Toisessa testissä on kolme EVSE-tolppaa, joita kaikkia on rajoitettu eri verran. Oikein toimiessaan algoritmi poistaa rajoituksia suhteessa eniten siltä, jota on eniten rajoitettu. Algoritmin tulee oikein toimiessaan poistaa rajoituksia yhteensä maksimissaan vain maksiminosto-muuttujan verran.

Taulukko 6. Testit kuudennen vaatimuksen testaamiseen.

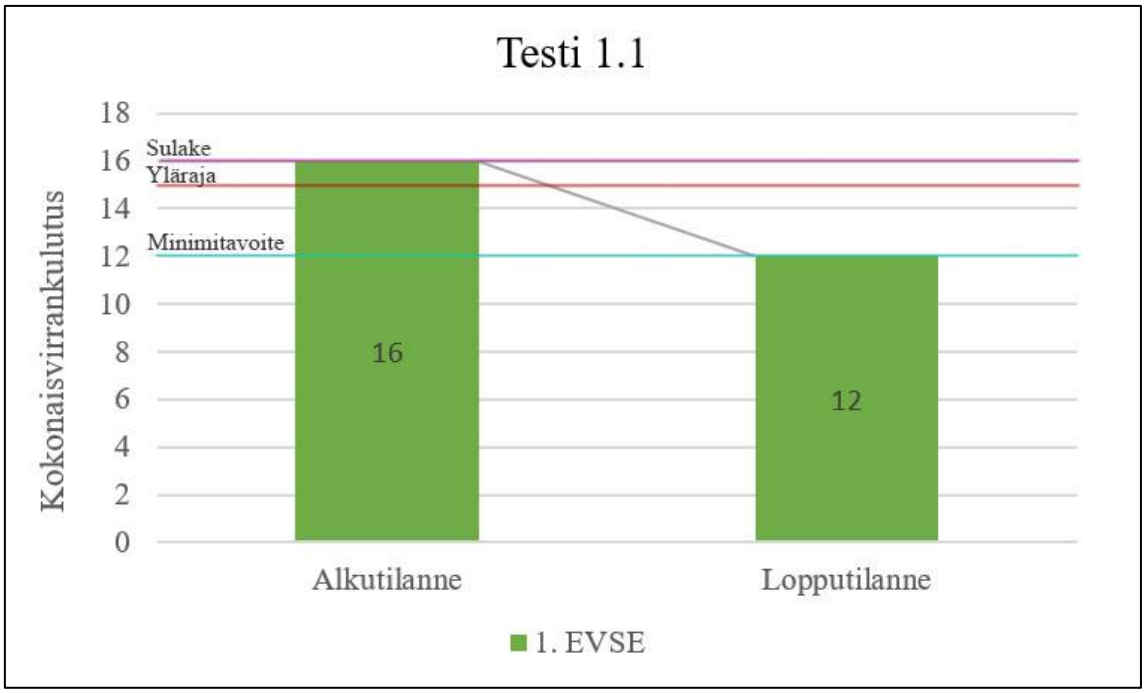
Nro.	Muuttujat	Kokoonpano	Testitapaus	Odotettu tulos
6.1	Alapuskuri: 12 Maksiminosto: 4	Sulake: 32A 1. EVSE: 6A (rajoitus: 10A) 2. EVSE: 6A (rajoitus: 10A)	Kaksi rajoitettua EVSE-tolppaa, joiden virrankulutus on yhteensä 12 ampeeria.	Molempien tolppien rajoituksia poistetaan yhteensä 4 ampeeria, molemmilta tolpilta 2 ampeeria. Virrankulutus nousee 12 ampeerista 16 ampeeriin.
6.2	Alapuskuri: 5 Maksiminosto: 8	Sulake: 40A 1. EVSE: 10A (rajoitus: 6A) 2. EVSE: 8A (rajoitus: 8A) 3. EVSE: 6A (rajoitus: 10A)	Kolme EVSE-tolppaa, joita kaikkia on rajoitettu eri määrä.	Ensimmäisen ja toisen EVSE-tolpan rajoitusta poistetaan 2 ampeeria, kolmannen EVSE-tolpan rajoitusta poistetaan 3 ampeeria.

5.2 Testit

Seuraavaksi käydään läpi vaatimusten yhteydessä määritetyt testit. Testit kuvataan kaavion avulla, missä kaavion ensimmäinen palkki on alkutilanne ja toinen palkki lopputilanne. Y-akseli kuvaa vaiheen kokonaisvirrankulutusta. Akselille on myös piirretty eri väreillä testien muuttujien kautta tulevat rajat ja arvot. Työn liitteenä (liite 1) on algoritmin testien tulosteet.

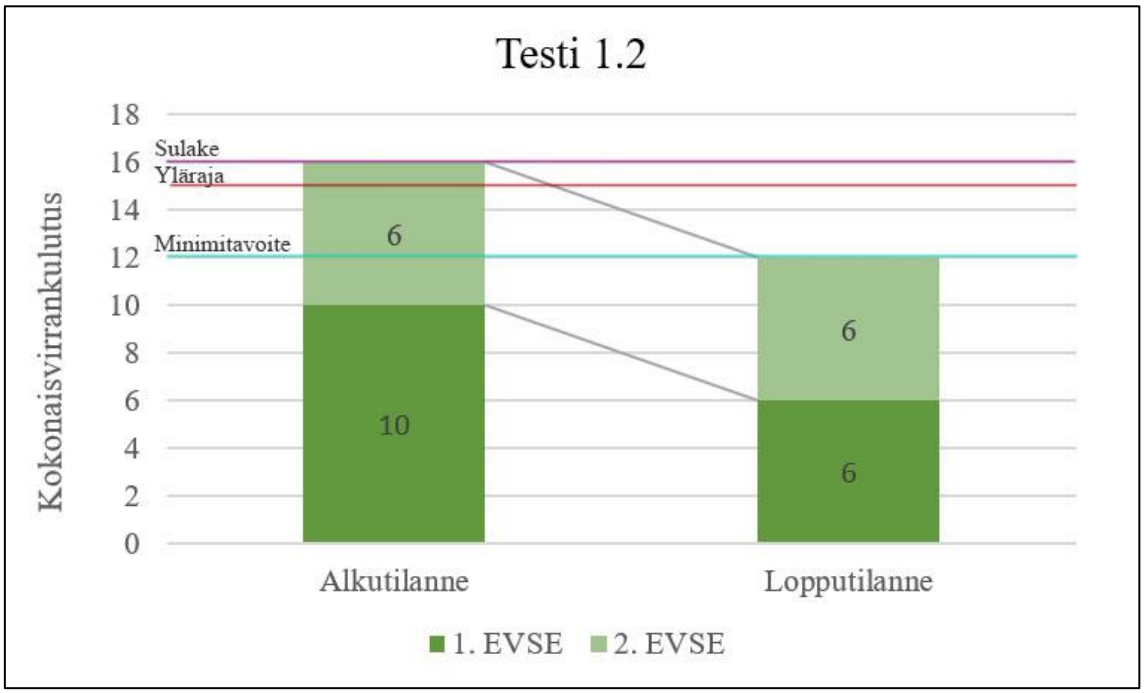
5.2.1 Ensimmäisen vaatimuksen testit

Ensimmäinen vaatimus on, että algoritmi osaa laskea kulutusta rajoittamalla EVSE-tolppaa. Kuvassa 33 on kuvattu ensimmäisen vaatimuksen ensimmäisen testin alku- ja lopputilanne. Testissä testattiin yhden EVSE-tolpan rajoittamista. Odotettu tulos oli, että latauslaitetta rajoitetaan 16 ampeerista 12 ampeeriin eli minimilaskun verran. Odotettu tulos toteutui, koska tolppaa rajoitettiin juurikin 4 ampeeria, eli algoritmi läpäisee testin 1.1.



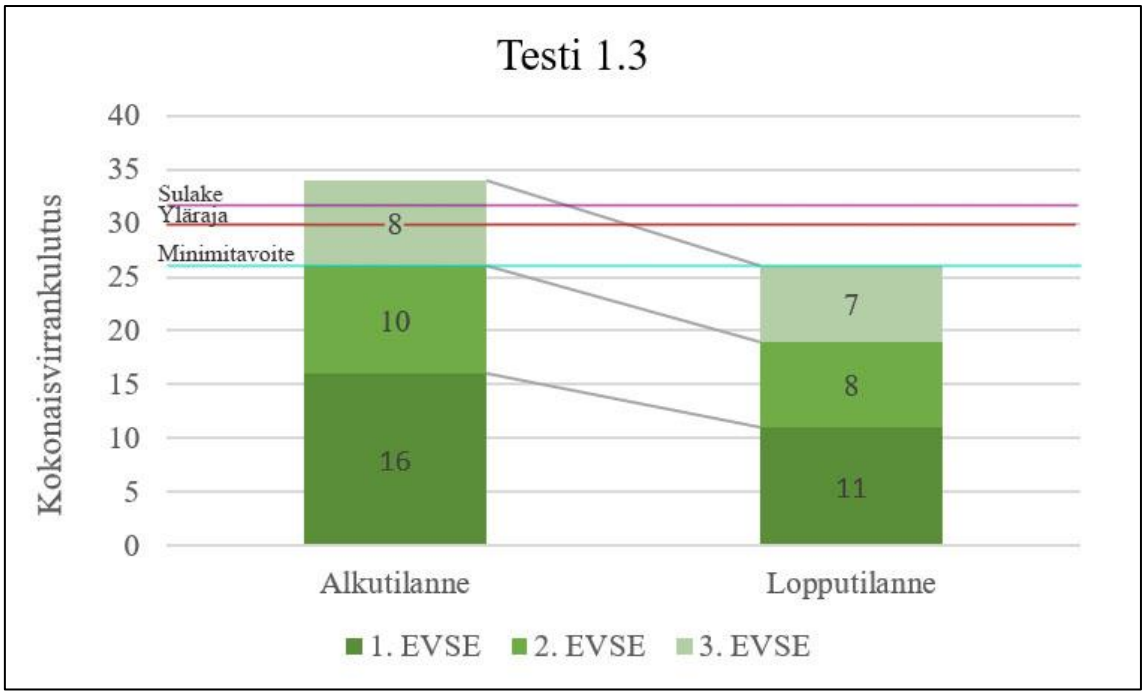
Kuva 33. Testin 1.1 alku- ja lopputilanne.

Kuvassa 34 on kuvattu toinen ensimmäisen vaatimuksen testeistä. Testissä testattiin virrankulutuksen rajoittamista, kun on kaksi latauslaitetta, joista toinen on jo rajoitettu minimiin. Odotettu tulos oli, että ensimmäinen EVSE-tolppa rajoitetaan 10 ampeerista 6 ampeeriin, jolloin kokonaiskulutus laskee 16 ampeerista 12 ampeeriin. Algoritmi toimi oikein ja läpäisee testin 1.2.



Kuva 34. Testin 1.2 alku- ja lopputilanne.

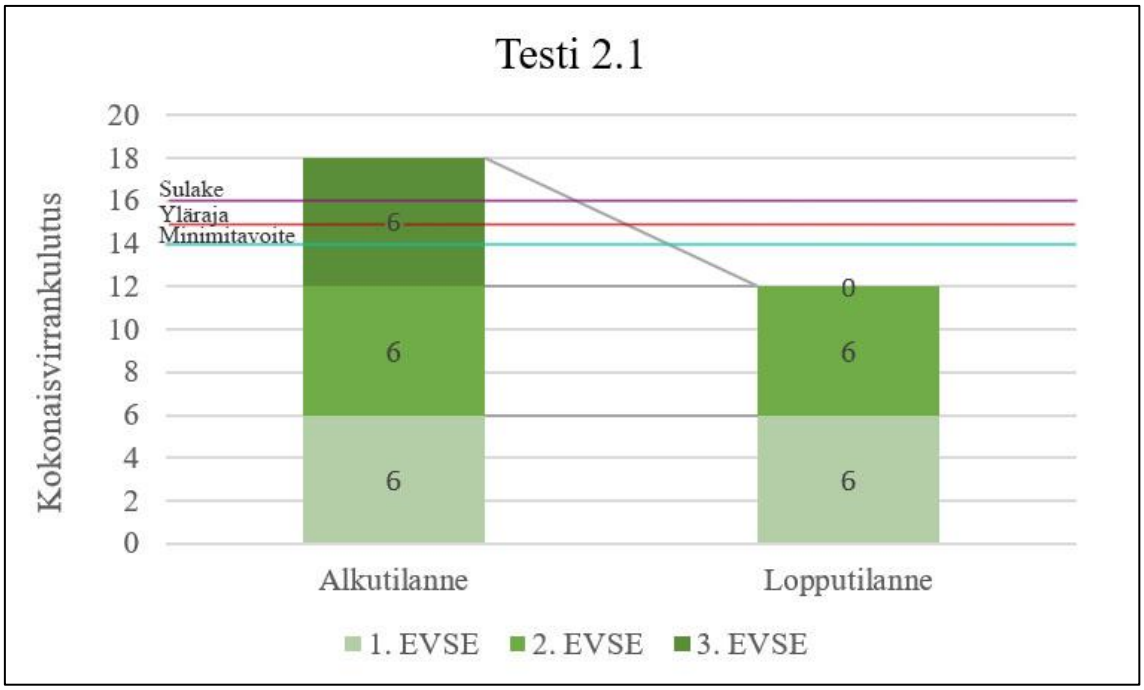
Kolmas testi ensimmäiselle vaatimukselle oli kolmen latauslaitteen rajoittaminen, kun latauslaitteet kuluttavat eri määrän virtaa. Testin tulos on esitetty kuvassa 35. Odotettu tulos testille oli, että kokonaisvirrankulutus laskee 34 ampeerista 26 ampeeriin. Ensimmäistä EVSE-tolppaa rajoitetaan 5 ampeeria, toista 2 ampeeria ja kolmatta 1 ampeeri. Odotettu tulos toteutui, eli algoritmi läpäisee testin 1.3.



Kuva 35. Testin 1.3 alku- ja lopputilanne.

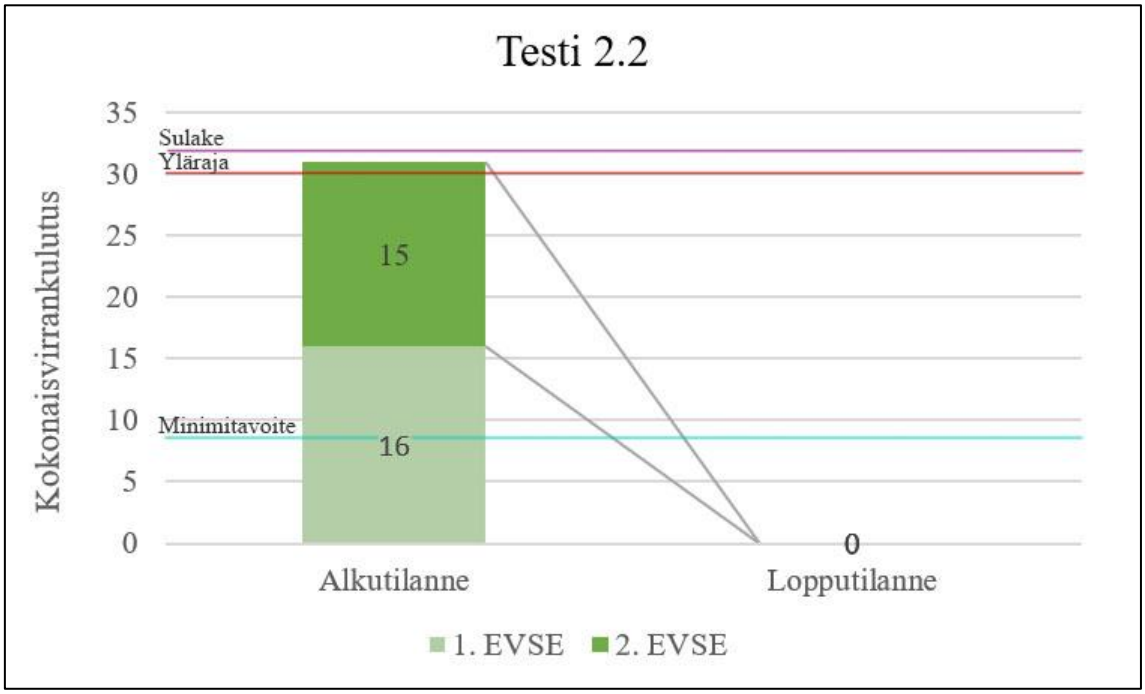
5.2.2 Toisen vaatimuksen testit

Toinen vaatimus oli, että algoritmi osaa laskea kulutusta sammuttamalla EVSE-tolppia. Kuvassa 36 on esitetty ensimmäisen testin tulos. Odotettu tulos testille oli, että algoritmi sammuttaa pisimpään päällä olleen EVSE-tolpan ja kokonaisvirrankulutus laskee 18 ampeerista 12 ampeeriin. Algoritmi osasi sammuttaa oikean tolpan ja odotettu tulos toteutui. Kuvassa ei näy mikä tolpista on ollut pisimpään päällä, mutta asian voi tarkistaa työn liitteenä (liite 1) olevista testien tulosteista. Algoritmi läpäisee testin 2.1.



Kuva 36. Testin 2.1 alku- ja lopputilanne.

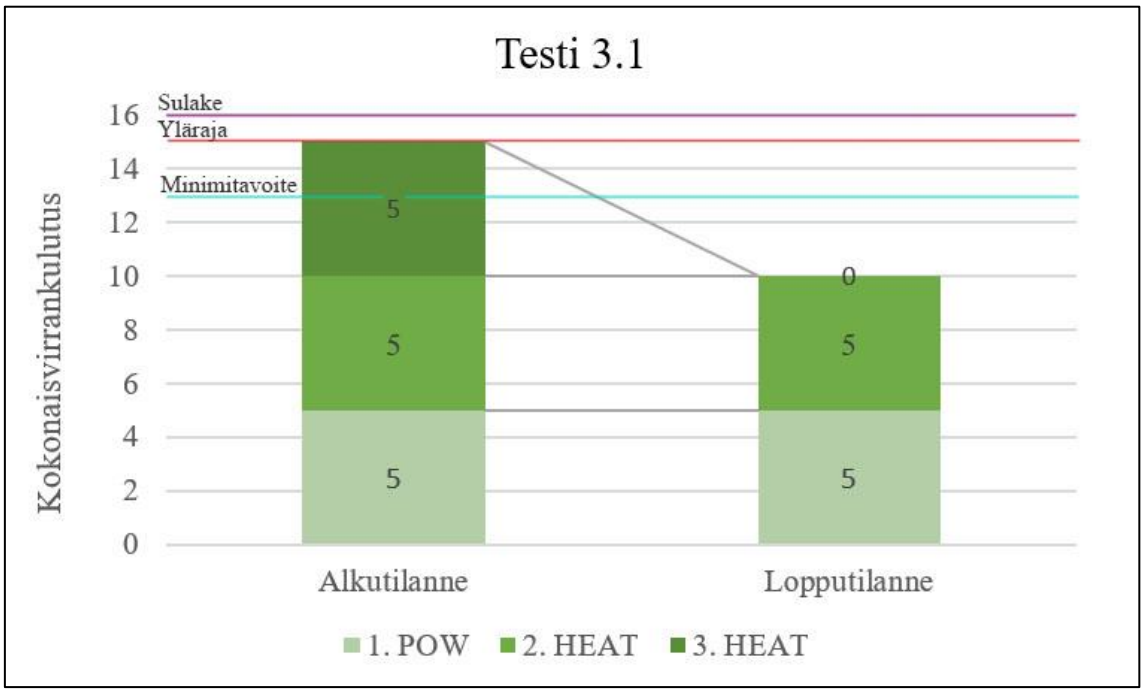
Toisen testin testitapauksessa oli kaksi latauslaitetta, jotka käyttävät yhteensä 31 ampeeria virtaa. Odotettu tulos oli, että kokonaisvirrankulutus laskee 31 ampeerista 8 ampeeriin. Toinen latauslaite sammutetaan ja toinen rajoitetaan 8 ampeeriin. Kuvassa 37 on kuvattu testin tulos. Algoritmi sammuttaa molemmat latauslaitteet, eli ei toimi oikein. Testi 2.2 on hylätty.



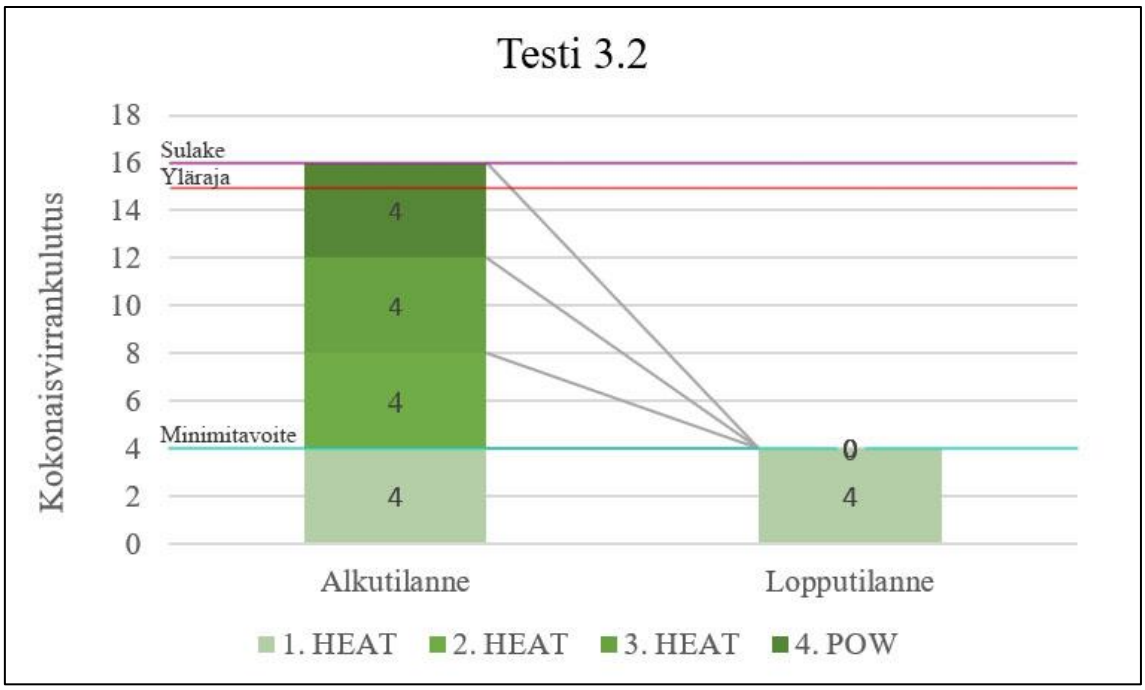
Kuva 37. Testin 2.2 alku- ja lopputilanne.

5.2.3 Kolmannen vaatimuksen testit

Kolmas vaatimus oli, että algoritmi osaa laskea virrankulutusta sammuttamalla POW- ja HEAT-tolppia. Ensimmäisessä testissä oli kolme POW- ja HEAT-tolppaa, joista algoritmin tulisi sammuttaa pisimpään päällä ollut. Kuvassa 38 on kuvattu testin tulos. Algoritmi osasi sammuttaa pisimpään päällä olleen tolpan. Liitteenä olevista testien tulosteista (liite 1) voi tarkistaa, että pisimpään päällä ollut tolppa sammui, eli testi 3.1 on läpäisty.



Kuva 38. Testin 3.1 alku- ja lopputilanne.

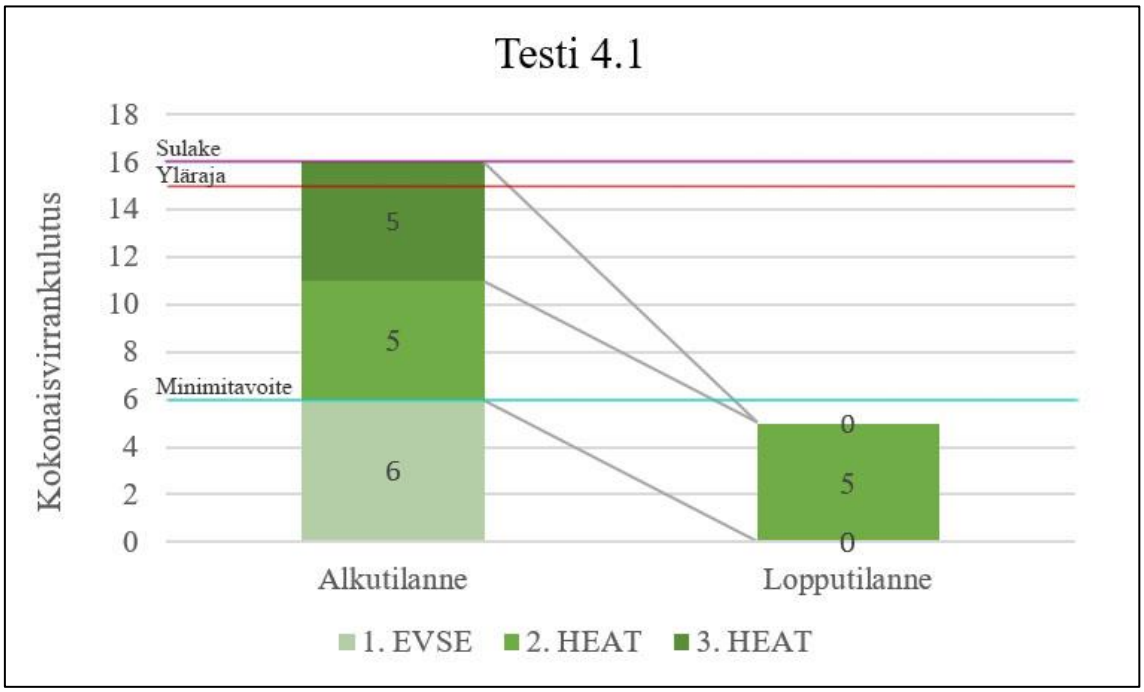


Kuva 39. Testin 3.2 alku- ja lopputilanne.

Kuvassa 39 on kuvattu toinen testi vaatimukselle. Odotettu tulos on, että kolme tolpista sammutetaan käynnistysjärjestyksen mukaan. Odotettu tulos toteutui, sillä kolme tolppaa sammutetiin ja lyhyimmän aikaa päällä ollut jäi päälle. Käynnistymisajat ovat nähtävissä liitteessä 1. Algoritmi läpäisee testin 3.2.

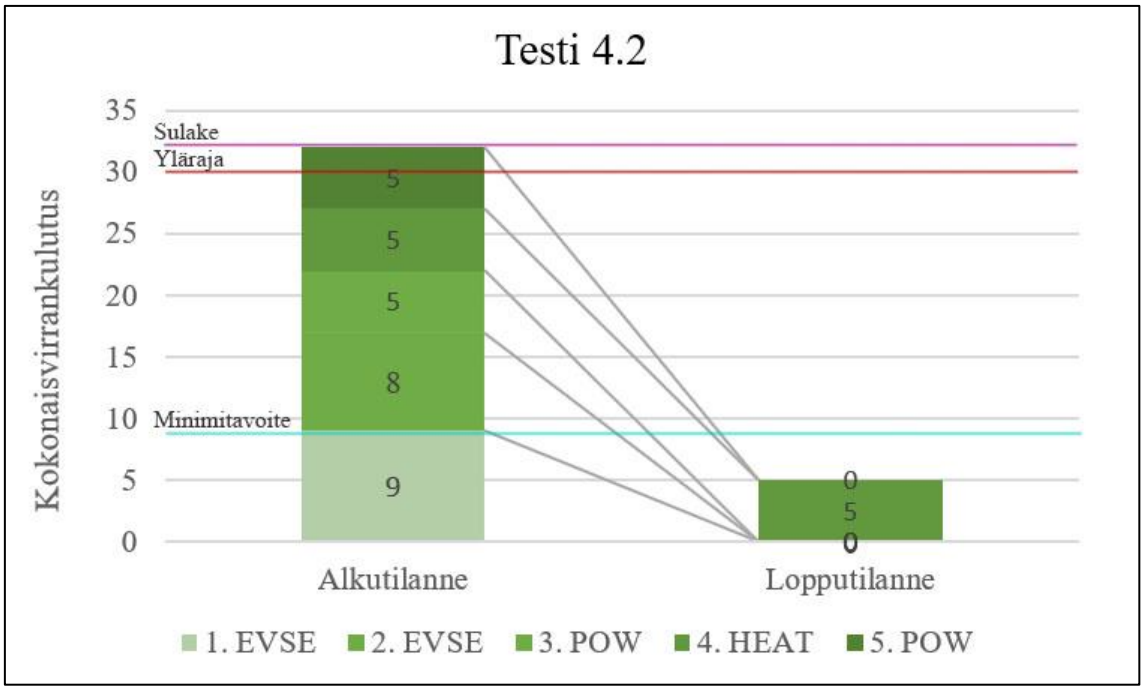
5.2.4 Neljännen vaatimuksen testit

Neljäs vaatimus oli, että algoritmi osaa laskea virrankulutusta sammuttamalla EVSE-, POW- ja HEAT-tolppia. Ensimmäisessä testissä algoritmin tulisi sammuttaa EVSE-tolppa sekä pisimpään päällä ollut HEAT-tolppa. Odotettu tulos toteutui (kuva 40 ja liite 1), joten algoritmi läpäisee testin 4.1.



Kuva 40. Testin 4.1 alku- ja lopputilanne.

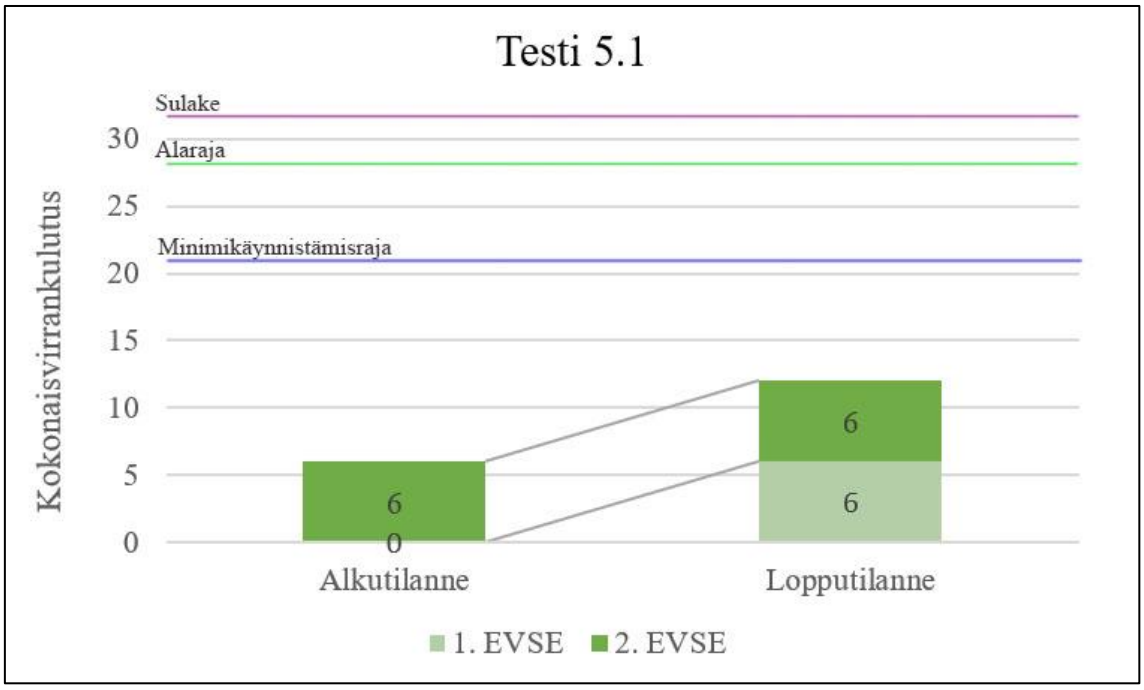
Toisessa testissä testattiin useamman EVSE-, POW ja HEAT-tolpan sammuttamista. Odotettu tulos testille oli, että molemmat EVSE-tolpat sammutetaan, jonka jälkeen sammutetaan kaksi pisimpään päällä olleista POW- ja HEAT-tolpista, jotta virrankulutus tippuu minimitavoitteen alapuolelle. Odotettu tulos toteutui (kuva 41 ja liite 1), eli testi 4.2 on läpäisty.



Kuva 41. Testin 4.2 alku- ja lopputilanne.

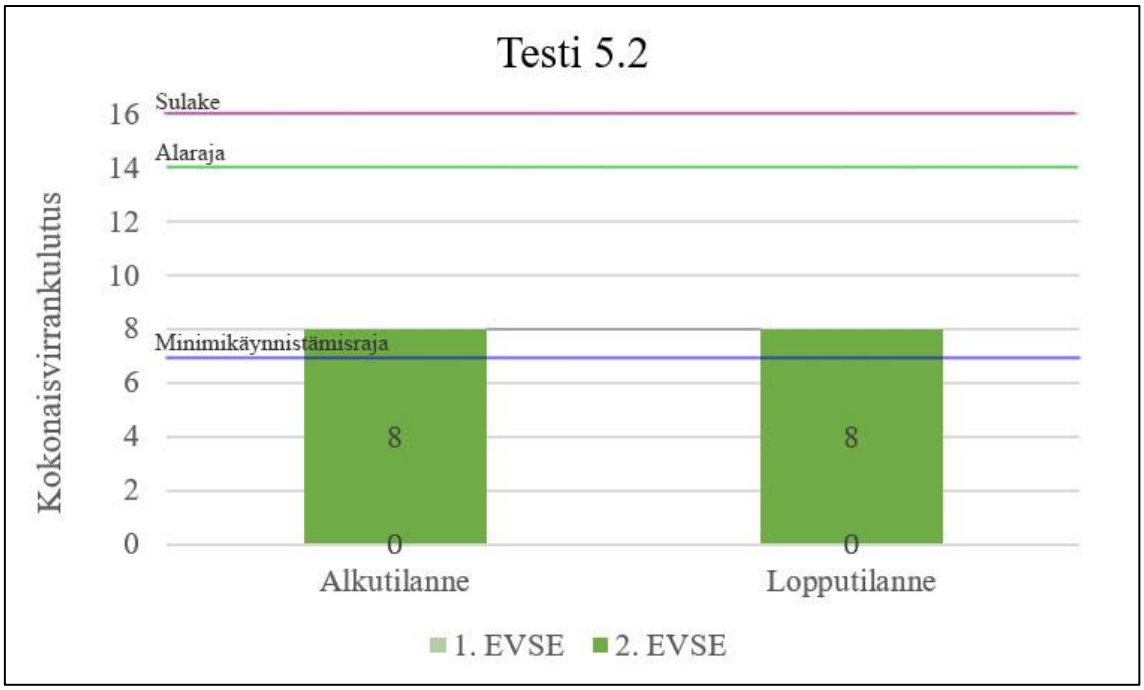
5.2.5 Viidennen vaatimuksen testit

Viides vaatimus oli, että algoritmi osaa nostaa virrankulutusta käynnistämällä EVSE-tolppia uudelleen. Ensimmäisessä testissä oli yksi sammutettu EVSE-tolppa ja odotettu tulos oli, että algoritmi käynnistää sammutetun tolpan ja asettaa sen virrankulutuksen 6 ampeeriin. Odotettu tulos toteutui (kuva 42), eli algoritmi läpäisee testin 5.1.



Kuva 42. Testin 5.1 alku- ja lopputilanne.

Vaatimuksen toisessa testissä testattiin tilannetta, jossa virrankulutus on minimikäynnistysrajan yläpuolella. Algoritmin ei tulisi käynnistää sammutettua EVSE-tolppaa uudelleen, koska sille ei ole tarpeeksi tilaa. Odotettu tulos toteutui (kuva 43), koska algoritmi ei käynnistänyt latauslaitetta, eli algoritmi läpäisee testin 5.2.

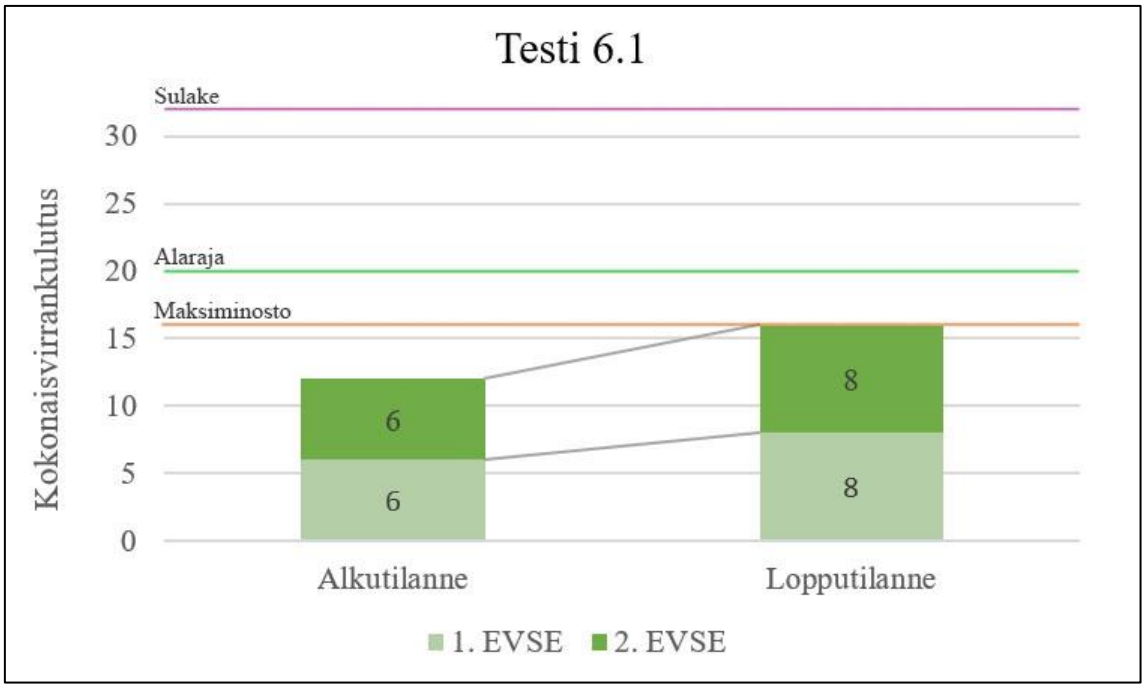


Kuva 43. Testin 5.2 alku- ja lopputilanne.

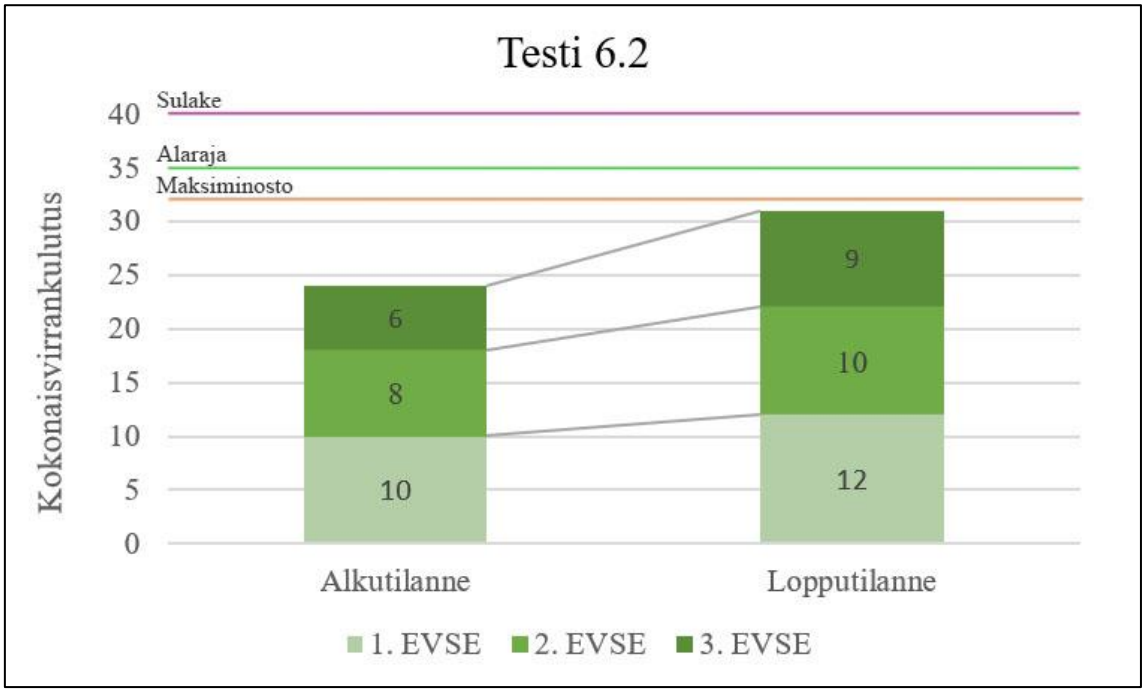
5.2.6 Kuudennen vaatimuksen testit

Kuudes ja viimeinen vaatimus oli, että algoritmi osaa nostaa vaiheen virrankulutusta poistamalla tolppien rajoituksia. Ensimmäisessä testissä oli kaksi EVSE-tolppaa, jotka oli rajoitettu minimiin eli 6 ampeeriin. Odotettu tulos oli, että molempien rajoituksia poistetaan 2 ampeeria eli yhteensä 4 ampeeria, jolloin kokonaisvirrankulutus nousee 12 ampeerista 16 ampeeriin. Kuvassa 44 on esitetty testin alku- ja lopputilanne. Virrankulutus nostettiin maksiminostoon asti. Odotettu tulos toteutui, eli algoritmi läpäisee testin 6.1.

Toisessa testissä testattiin rajoitusten poistoa tilanteessa, jossa vaiheella on useita EVSE-tolppia, jotka kaikki kuluttavat eri määrän virtaa. Odotettu tulos oli, että ensimmäisen ja toisen EVSE-tolpan rajoitusta poistetaan 2 ampeeria ja kolmannen EVSE-tolpan rajoitusta poistetaan 3 ampeeria. Odotettu tulos toteutui (kuva 45), eli algoritmi läpäisee testin 6.2. Kuvassa näkyy hyvin, ettei rajoituksia poisteta maksiminostoon asti, vaikka teoriassa se olisi mahdollista. Syy tähän on, että laskuista tulee murtolukuja ja nämä luvut pyöristetään alaspäin, sillä latauslaitteen virrankulutuksen täytyy olla kokonaisluku.



Kuva 44. Testin 6.1 alku- ja lopputilanne.



Kuva 45. Testin 6.2 alku- ja lopputilanne.

5.3 Testien yhteenveto

Seuraavaksi tehdään yhteenveto testien tuloksista ja algoritmin toiminnasta testien aikana. Kaikkien testien tulokset ja selitteet on koottu taulukkoon 7.

Algoritmi selviytyi testeistä hyvin, sillä vain yksi testeistä ei tuottanut odotettua tulosta. Testin 2.2 tulos hylättiin, sillä algoritmi sammutti molemmat EVSE-tolpat, vaikka

vain toisen latauslaitteen sammuttaminen ja toisen latauslaitteen rajaaminen olisi riittänyt. Todellisuudessa tämä on harvinainen tilanne, sillä yleensä yhden latauslaitteen sammuttamisesta seuraava minimissään 6 ampeerin virrankulutuksen lasku riittää hyvin.

Testin 2.2 ongelmat johtuvat siitä, että tällaista vaihtoehtoa ei ole otettu huomioon toteutuksessa. Jos katsomme vuokaaviota kuvassa 30, huomaamme kohdissa 35-39, että algoritmi on toteutettu niin, että se sammuttaa latauslaitteita, kunnes virrankulutusta on laskettu tarpeeksi. Ongelmanydin on kohdassa 31, jossa lasketaan riittääkö EVSE-tolppien rajoittaminen vai pitääkö niitä sammuttaa. On totta, että EVSE-tolppien pelkkä rajoittaminen ei riitä, mutta algoritmi ei ota huomioon tilannetta, jossa riittäisi esimerkiksi yhden EVSE-tolpan sammuttaminen ja toisen rajoittaminen, kuten testissä 2.2 tilanne oli.

Algoritmi alun perin toteutettiin sen mukaan, että EVSE-tolppia voisi rajoittaa nollaan ampeeriin asti. Myöhemmin selvisi, että 6 ampeeria on minimivirrankulutus. Tämän tiedon jälkeen algoritmin logiikkaa korjailtiin ja toteutettiin uudelleen, minkä seurauksena tämä skenaario on unohtunut toteutuksesta.

Taulukko 7. Yhteenveto testeistä ja niiden tuloksista.

Vaatimus	**Nro.**	**Tulos**	**Selitys**
Vaatimus 1	1.1	Ok	Odotettu tulos toteutui.
	1.2	Ok	Odotettu tulos toteutui.
	1.3	Ok	Odotettu tulos toteutui.
Vaatimus 2	2.1	Ok	Odotettu tulos toteutui.
	2.2	Hylätty	Algoritmi sammutti molemmat latauslaitteet, vaikka olisi riittänyt vain toisen latauslaitteen sammuttaminen. Toista latauslaitetta olisi pitänyt rajoittaa 8 ampeeriin sammuttamisen sijasta.
Vaatimus 3	3.1	Ok	Odotettu tulos toteutui.
	3.2	Ok	Odotettu tulos toteutui.
Vaatimus 4	4.1	Ok	Odotettu tulos toteutui.
	4.2	Ok	Odotettu tulos toteutui.
Vaatimus 5	5.1	Ok	Odotettu tulos toteutui.
	5.2	Ok	Odotettu tulos toteutui.
Vaatimus 6	6.1	Ok	Odotettu tulos toteutui.
	6.2	Ok	Odotettu tulos toteutui.

Algoritmi täyttää vaatimuksista viisi kuudesta (taulukko 8). Vaatimus 2 ei täyty testin 2.2 epäonnistumisen takia. Sinänsä algoritmi toimii testin 2.2 kohdalla täysin oikein eli kuten se on koodattu, mutta sen logiikassa on vain suunnitteluvirhe.

Taulukko 8. Yhteenveto vaatimuksista ja tuloksista.

Vaatimus	**Tulos**
Vaatimus 1: Algoritmi osaa laskea kulutusta rajoittamalla EVSE-tolppia.	Ok
Vaatimus 2: Algoritmin osaa laskea kulutusta sammuttamalla EVSE-tolppia.	Hylätty
Vaatimus 3: Algoritmi osaa laskea kulutusta sammuttamalla POW ja HEAT-tolppia.	Ok
Vaatimus 4: Algoritmi osaa laskea kulutusta sammuttamalla EVSE, POW ja HEAT-tolppia.	Ok
Vaatimus 5: Algoritmi osaa nostaa kulutusta käynnistämällä EVSE-tolppia.	Ok
Vaatimus 6: Algoritmi osaa nostaa kulutusta poistamalla EVSE-tolppien rajoituksia.	Ok

Testien tuloksista voidaan myös tehdä muitakin johtopäätöksiä, kuten esimerkiksi se, että algoritmi osaa tulkita oikein, milloin vaiheen kokonaisvirrankulutusta tulee laskea ja milloin nostaa. Tämän lisäksi algoritmi osaa myös priorisoida toimenpiteet oikein, eli se tekee ensin mahdolliset toimenpiteet EVSE-tolpille ja vasta sen jälkeen sammuttaa POW- ja HEAT-tolppia. Algoritmi osaa siis aina valita oikean toimenpiteen, vaikkakin yksi toimenpiteistä, eli EVSE-tolppien sammuttaminen, toimii hieman väärin. Myös latauslaitteiden rajoituksia lisättiin ja poistettiin laskukaavojen mukaisesti maksiminosto- ja minimilaskurajat huomioiden. Tämä on tärkeää, sillä algoritmin tulee osata totella sille asetettuja rajoja, jotta esimerkiksi kulutusta ei nosteta liian paljoa kerralla. Muuttujat siis toimivat täysin oikein testeissä. Tämän lisäksi algoritmi osaa myös valita sammutettavat tolpat käynnistysajan mukaan aloittaen pisimpään päällä olleesta tolpasta. Algoritmin käyttämä tietorakenne ja vaihe- ja tolppaoliot toimivat myös oikein, sillä kaikki suoritetut testit käyttivät niitä useissa eri välivaiheissa. Tästä kaikesta voidaan tehdä johtopäätös, että algoritmin toiminta on lähes kokonaan etukäteen ennustettavissa ja lähes oikeellista.

Testien suoritusajat löytyvät liitteenä olevien testien komentoikkunatulosteiden yhteydestä. Kaikki suoritusajat olivat 13 ja 53 millisekunnin välillä, mutta testit tulostavat komentoikkunaan paljon erilaisia tekstirivejä, mikä saattaa vääristää suoritusaikoja, sillä virallinen versio ei näitä tulosta komentoikkunaan. Suoritusajat myös vaihtelivat suuresti

jokaisella ajokerralla samankin testin kohdalla, joten niistä on vaikea tehdä suurempia johtopäätöksiä.

Tässä luvussa tehdyt testit eivät kattaneet aivan kokonaan algoritmin toimintaa. Testien ulkopuolelle jääviä toimintoja on viestijonon kuunteleminen, viestien lähettäminen tolpille, tietokannasta tiedon hakeminen ja tallentaminen muistirakenteeseen, raakatiedon lukeminen muistiin ja tallentaminen tietokantaan. Testit oli suunnattu enemmänkin testaamaan algoritmin oikeellista toimintaa kohdan 3.5 määritelmien mukaisesti. Algoritmia tulisi myös testata laajemmin erilaisilla kokoonpanoilla. Näissä testeissä ei esimerkiksi testata ollenkaan, miten algoritmi toimii, kun latauslaitteita on 20, 50 tai 100.

Viestijonon kuuntelemista, tietokannasta tiedon hakemista ja tallentamista muistirakenteeseen ja raakatiedon viemistä tietokantaan testattiin reilusti algoritmin toteutuksen yhteydessä. Kokonaiskuvassa algoritmin toiminnan kannalta tärkeintä on kuitenkin se, että algoritmin virrankulutusta säätelevä osa toimii oikeellisesti ja sen vuoksi tämän luvun testit käsittelivät tätä osuutta algoritmista. Tämän lisäksi edellä mainittujen toimintojen testaaminen on myös vaikeampaa käytännössä.

Viestin lähettämistä tolpille testattiin myös oikeiden laitteiden kanssa. HEAT-tolppaan kytkettiin kiinni led-lamppu, joka sammui aina kun algoritmi sammutti kyseisen tolpan. EVSE-tolpan toimintaa testattiin oikealla latauslaitteella, johon oli kiinnitetty latashybridiauto. Tälle latauslaitteelle lähetettiin viestejä ja tarkkailtiin latauslaitteen ja auton toimintaa. Latauslaite sai viestit ja toimi viestin komennon mukaisesti. Kaikki komennot toimivat suunnitellusti ja auto totteli latauslaitteen mukana. Seuraavassa luvussa analysoidaan tarkemmin kuormantasausalgoritmia ja sen toimintaa.

6 Kuormantasausalgoritmin analysointi

Seuraavaksi analysoidaan algoritmin toimintaa tarkemmin. Aluksi kohdassa 6.1 pohditaan mahdollisia algoritmiin liittyviä ongelmia sekä ratkaisuja näihin. Kohdassa 6.2 käsitellään algoritmin ratkaisun skaalautuvuutta. Kohdassa 6.3 pohditaan algoritmille muita mahdollisia käyttökohteita. Kohdassa 6.4 tehdään katsaus ja vertailu muihin markkinoilla oleviin kuormantasausratkaisuihin.

6.1 Mahdolliset ongelmat ja ratkaisuehdotukset

Algoritmin toimintaan liittyy useita mahdollisia ongelmia ja tilanteita, joita tässä toteutuksessa ei välttämättä ole otettu huomioon. Osa ongelmista saattaa ilmetä vasta kun algoritmi otetaan pidempiaikaisesti testikäyttöön. Tässä kohdassa kuitenkin pohditaan mahdollisia ongelmia, joita nykyinen toteutus ei välttämättä ota huomioon. Ongelmien kanssa samassa yhteydessä esitetään ratkaisuja, joilla ongelmista selvittäisiin.

Virhetilanteen sattuessa on ehdottoman tärkeää, että algoritmi raportoi selvästi, millainen virhe on tapahtunut ja missä kohtaa algoritmin suoritusta. Tämän pohjalta ongelmaa on helpompi lähestyä ja toteuttaa ratkaisu nopeammin. Tällä hetkellä algoritmi tulostaa virhetilanteet Azureen WebJobin lokiin ja jatkaa normaalisti toimintaa eteenpäin. Tämän lisäksi kaikista algoritmin tekemistä toimenpiteistä tulee jäädä jälki tietokantaan, jotta niitä voidaan jälkikäteen tarkistaa ja seurata. Tällä hetkellä kaikki tolpalta tuleva viestiliikenne tallennetaan tietokantaan ja näissä viesteissä myös näkyy algoritmin lähettämät komennot.

Algoritmin huoltaminen ja päivittäminen on suhteellisen helppoa, sillä sitä on mahdollista kehittää ja testata omassa työympäristössä ilman, että oikeaa algoritmia pyörittävää Azuren WebJobia tarvitsisi pysäyttää. Kun uusi korjattu versio algoritmista on valmis, se voidaan päivittää pyörimään Azuren WebJobin tilalle. Tästä saattaa seurata hetkellinen katkos, kun vaihdos tehdään, mutta on myös mahdollista lisätä useampi Azure WebJob pyörimään rinnakkain, jolloin tällaista katkosta ei synny.

6.1.1 Sammuttaminen

Algoritmin sammuttaminen väärällä hetkellä saattaa aiheuttaa ongelmia, kuten esimerkiksi tiedon menettämistä ja keskenjääneitä toimenpiteitä. Paras kohta Azure WebJobin sulkemiseen on juuri sen jälkeen, kun algoritmi on tallettanut raakatiedon tietokantaan. Tämä toimenpide tehdään 30 sekunnin välein. Tällöin vältytään raakatiedon menettämiseltä ja keskenjääneiltä toimenpiteiltä, sillä silloin yhdenkään viestin käsittely ei ole kesken. Algoritmin sammuttaminen kesken kuormantasausoperaation saattaa esimerkiksi johtaa siihen, että vain osaa vaiheen tolpista rajoitetaan ja tilanne jää kesken. Tällaisen sammuttamisen ajoittaminen on kuitenkin käytännössä mahdotonta, sillä algoritmi toimii niin nopeasti.

Suurin puute algoritmin tämän hetkisessä toteutuksessa on, että tietokantaan ei tallenneta tietoa algoritmin tekemistä tolppien rajoituksista ja sammutuksista, vaan nämä tiedot ovat vain algoritmin muistissa tietorakenteessa. Käytännössä nämä tiedot menetetään kokonaan, mikäli algoritmi sammutetaan tai mikäli se käynnistyy uudelleen. Näiden tietojen menettämisestä seuraa, ettei algoritmi enää tiedä, mitä tolpista on rajoitettu tai sammutettu algoritmin toimesta ja mitkä vain lataavat pienemmällä teholla tai missä ei ole autoa ollenkaan latautumassa. Aikaisemmalla algoritmin ajokerralla rajoitetut tolpat jäisivät ikuisiksi ajoiksi rajoitetuiksi. Tämä ongelma täytyy korjata ennen kuin algoritmi otetaan laajemmin testikäyttöön. Yksi vaihtoehto ongelman ratkaisemiseen olisi, että aina samalla, kun raakatietoa tallennetaan kantaan 30 sekunnin välein, vietäisiin myös samalla tietokantaan tiedot tolppien rajoituksista ja sammutetuista tolpista. Tässä yhteydessä tulee huomioida sama kuin raakatiedon kanssa eli se, ettei tallentamattomia tietoja menetetä silloin kun ohjelma sammutetaan tietokantaan vientien välissä. Toinen vaihtoehto olisi viedä tiedot tietokantaan aina, kun tolppia rajoitetaan tai sammutetaan, mutta tämä ei ole läheskään niin tehokasta ja hidastaisi algoritmin toimintaa. Tämän vuoksi 30 sekunnin välein tapahtuva tiedon päivittäminen tietokantaan on mielestäni järkevin ratkaisu.

Kaikki edellä mainitut ongelmat liittyen tiedon menettämiseen ja toimenpiteiden keskeytymiseen sammuttamisen yhteydessä voidaan ratkaista toteuttamalla funktio, joka vie tietokantaan tallentamatta jääneet tiedot ja jää odottamaan WebJobin sulkeutumista. Tässä voidaan hyödyntää ”Cancellation token”-parametriä WebJobissa, joka mahdollistaa ohjelman sulavan sammumisen (graceful shutdown) [Microsoft 2018b]. Vaikka periaatteessa riittäisi, että tieto tolppien rajoituksista ja sammutetuista tolpista vietäisiin kantaan WebJobin sammumisen yhteydessä, kannattaa silti varautua siihen, ettei WebJob aina sammu sulavasti, vaan joskus saattaa myös tapahtua kova sammuminen (hard shutdown), jolloin kaikki tiedot menetettäisiin kokonaan. Tämän vuoksi on tärkeää, että pääasiassa tallentamattomat tiedot viedään tietokantaan 30 sekunnin välein ja WebJobin sammumisen yhteydessä tallennetaan vain mahdollisesti näiden tallennusten välissä tallentamatta jääneet tiedot.

Edellä mainittujen parannuksien jälkeen suurimmat algoritmin sammumiseen ja sammuttamiseen liittyvät ongelmat korjaantuvat, jolloin tältä osin algoritmi olisi valmis käyttöönotettavaksi.

6.1.2 Viestiliikenne

Viestiliikenteeseen liittyy useita mahdollisia ongelmia. Ongelmia voi olla tolpan päässä, jolloin tolppa menettää esimerkiksi internetyhteyden, eikä enää saa lähetettyä viestejä. Tällöin algoritmi menettää tiedon tolpan virrankulutuksesta, jolloin se ei myöskään osaa laskea sitä mukaan vaiheen kokonaisvirrankutukseen. Tästä voisi esimerkiksi seurata tilanne, jossa algoritmi ei tiedä ylärajan oikeasti ylittyvän, koska yhden tolpan virrankulutus puuttuu kokonaan.

Jos olisi mahdollista mitata sähkökaapilta suoraan vaiheen kokonaisvirrankulutusta ja toimittaa tämä tieto algoritmille, edellä kuvatulta ongelmalta vältyttäisiin osittain. Tämän tyyppinen toteutus tullaan todennäköisesti tulevaisuudessa myös lisäämään algoritmiin, jonka jälkeen algoritmi saisi viestejä sekä tolpilta että sähkökaapilta. Algoritmi kuitenkin tarvitsee myös tiedon tolppien virrankulutuksesta, jotta se tietää mitä sen pitää rajoittaa tai sammuttaa.

Toinen mahdollinen ongelma viestiliikenteeseen liittyen on, ettei tällä hetkellä varmisteta mitenkään sitä, menevätkö algoritmin lähettämät viestit perille asti tolpalle. Tällä hetkellä oletetaan, että kaikki algoritmin lähettämät viestit menevät perille ja tolppa toimii viestin komennon mukaisesti. Todennäköisesti tätä ongelmaa kannattaisi lähteä korjaamaan sitä kautta, että viestiliikenteessä olisi käytössä pyyntö-vastaus (request-response) malli, jossa algoritmi saisi tolpalta vastauksen, joka vahvistaisi komennon läpimenon.

Kolmas mahdollinen ongelma viestiliikenteeseen liittyen on, että jos algoritmi saa jostain syystä viestejä myöhässä, niin myös päätöksiä tehtäisiin vanhentuneen tiedon pohjalta. Tämän vuoksi on tärkeää, ettei välittäjänä toimiva RabbitMQ säilö jumiutumistilanteessa viestejä jonoon, josta algoritmille toimitettaisiin vanhentuneita viestejä. Tältä ongelmalta voi myös välttyä lisäämällä EVSE-tolpilta tuleviin viesteihin aikaleima, jonka perusteella voidaan katsoa koska viesti on lähtenyt latauslaitteelta. POW- ja HEAT-tolpissa tämä aikaleima jo löytyy.

Näkisin, etteivät tässä alakohdassa esitetyt viestiliikenteeseen liittyvät ongelmat estä mitenkään algoritmin ensimmäisen version testaamista käytännössä. Todennäköisesti nämä ovat juurikin sellaisia ongelmia, jotka ilmenevät ja selkiytyvät paremmin sitten vasta kun algoritmia aletaan kokeilla todellisuudessa.

6.1.3 Reagointinopeus

Kolmas ongelma liittyy algoritmin reagointinopeuteen. Tällä hetkellä tolpat lähettävät 30 sekunnin välein virrankulutusviestejä, joiden pohjalta algoritmi toimii. Mikäli tällä välillä tapahtuu merkittäviä muutoksia vaiheen virrankulutuksessa, algoritmi ei saa tietoa siitä. Käytännössä ongelmia aiheuttavat muutokset liittyvät virrankulutuksen kasvuun. Virrankulutuksen laskeminen tällä välillä ei aiheuta ongelmia. Kun virrankulutus kasvaa niin paljon viestien välissä, että se ylittää sulakkeen rajan, sulake saattaa ehtiä palamaan ennen kuin algoritmi ehtii laskemaan vaiheen virrankulutusta.

Tämän ongelman voisi ratkaista sillä, että tolppa lähettäisi virrankulutusviestin aina kun tolpan virrankulutukseen tulee suurempia muutoksia. Tämä mahdollistaisi algoritmin nopean toiminnan, mutta algoritmin logiikkaa täytyy muuttaa, jos normaalien 30 sekunnin välein kulkevien virrankulutusviestien lähettäminen lopetettaisiin kokonaan. Tällä hetkellä vaiheen virrankulutuksen laskemiseen valitaan mukaan sellaiset tolpat, jotka ovat olleet aktiivisia eli lähettäneet virrankulutusviestin muuttujissa määritellyn ajan sisällä. Tällä tavoin jätetään huomioimatta vanhentuneet arvot, joita saattaa seurata, jos tolppa

esimerkiksi menettäisi verkkoyhteyden ja lopettaisi virrankulutusviestien lähettämisen. Mikäli muutoksista lähetettävät virrankulutusviestit tulevat normaalien virrankulutusviestien rinnalle, viestiliikenteen määrä hieman kasvaa, mutta algoritmin reagointinopeus kasvaa reilusti.

Reagointinopeuteen liittyvät ongelmat eivät mielestäni estä algoritmin ensimmäisen version käyttöönottoa oikeassa ympäristössä. Paremminkin käyttöönotosta voisi saada arvokasta tietoa siitä, miten nopeasti algoritmi ehtisi reagoimaan virrankulutuksen kasvamiseen ja kuinka usein oikeasti sulake pääsisi palamaan.

6.1.4 Käyttäjää ei tiedoteta

Tällä hetkellä käyttäjää ei tiedoteta mitenkään algoritmin tekemistä toimenpiteistä, joka tarkoittaa sitä, että esimerkiksi latauslaitteen käyttäjä saattaa ihmetellä miksi hänen autonsa latauslaite on sammutettu ja käynnistää sen itse uudelleen. Tämän vuoksi käyttäjälle tulisi lähettää ilmoitus kaikesta, mitä algoritmi tekee hänen käyttämilleen laitteille. Tämä vaatii sitä, että algoritmi tietää aina kenen auto tolppaan on kytketty, jotta tarvittavat tiedot saadaan välitettyä oikealle henkilölle. Tämä saattaa olla ongelma, jos kyseessä on julkisessa käytössä oleva tolppa, jonka käyttäjä vaihtuu. Tietokantaan tulisi aina tallentaa tieto käyttäjästä, joka tolpan käynnistää. Algoritmi päivittää tällä hetkellä tietokannasta tiedon tolpan käyttäjästä 30 sekunnin välein. Tämän lisäksi pitäisi vielä lisätä jokin tapa lähettää viestejä käyttäjän mobiililaitteelle algoritmin tekemistä muutoksista, jotta vältytään sekaannukselta. Käyttäjän tiedottamisen lisäksi voisi olla myös hyvä lisätä toiminto, joka estää sen, ettei käyttäjä voi itse käynnistää tolppaansa uudelleen sammuttamisen jälkeen ennen kuin vaiheella on taas tilaa tarpeeksi virrankulutuksessa. Olisi myös mahdollista toteuttaa ratkaisu, joka estäisi käyttäjää jo alun perin käynnistämästä tolppaa, mikäli vaiheen kokonaisvirrankulutuksessa ei ole enää tilaa.

Tämä ongelma ei estä algoritmin ottamista testikäyttöön, sillä testikäyttäjät voivat tarkkailla muuta kautta, kuten esimerkiksi WebJobin tulostelokista, miten algoritmi toimii. Toiminto käyttäjän tiedottamiseen kuitenkin vaaditaan ennen kuin algoritmi voidaan ottaa laajempaan käyttöön oikeiden asiakkaiden piiriin.

6.1.5 Muut ongelmat ja yleisiä kehityskohteita

Seuraavaksi käsitellään pienempiä ongelmia ja yleisiä kehityskohteita algoritmissa. Ensimmäinen näistä on, että tällä hetkellä kaikki käytetyt muuttujat ovat algoritmikohtaiset, joten kaikki vaiheet käyttävät samoja arvoja. Olisi toteutuksen kannalta järkevämpää, jos nämä tiedot vietäisiin tietokantaan vaiheen tauluun, josta jokaiselle vaiheelle olisi mahdollista määrittää esimerkiksi ylä- ja alapuskuriarvot, joiden pohjalta ylä- ja alarajat lasketaan. Testausmielessä algoritmikohtaiset globaalit muuttujat olivat helppoja säädellä ja vaihdella verrattuna siihen, että muutokset tulisi aina tehdä tietokantaan.

Algoritmiin voisi myös lisätä useamman erilaisen rajoitustavan, joka olisi vaihekohtainen. Tällä hetkellä tolppien rajoittamiseen on vain yksi tapa eli virrankulutuksen mukainen rajoittaminen tasaisesti kaikilta. Vaihtoehtoinen rajoitustapa voisi esimerkiksi olla pisimpään ladanneen tolpan sammuttaminen, jotta tilaa saadaan lisää. Useamman rajoitustavan lisääminen mahdollistaisi algoritmin toiminnan monipuolistumisen. Tällä hetkellä käytettävää rajoitustapaa ei ole testattu tilanteissa, joissa latauslaitteita on useita kymmeniä. Tällöin todennäköisesti algoritmi laskisi virrankulutusta liian paljon. Todennäköisesti suuriin parkkipaikkoihin kannattaisi soveltaa erilaista latauslaitteiden rajoitustapaa. Algoritmin ensimmäistä versiota tullaan testaamaan vain parkkipaikoissa, joissa on vain muutamia latauslaitteita, joten tämän ei pitäisi olla ongelma.

Algoritmin testeissä ilmennyt suunnitteluvirhe liittyen siihen, ettei algoritmi osaa sekä sammuttaa EVSE-tolppia että rajoittaa niitä samalla kerralla, tulisi myös korjata. Vastaavanlaisten ongelmien kannalta on kuitenkin tärkeää, että virheet tapahtuvat niin päin, että algoritmi laskee liian paljon kulutusta kuin jättää laskematta tarpeeksi. On myös harvinainen skenaario, että EVSE-tolpan sammuttaminen ei riitä saavuttamaan tarvittavaa virrankulutuksen laskua. Algoritmi olisi myös todennäköisesti seuraavalla ajokerralla käynnistänyt toisen sammutetuista EVSE-tolpista uudelleen. Silti mielestäni suunnitteluvirhe kannattaa korjata, sillä sen avulla voidaan välttyä turhalta EVSE-tolppien sammuttamiselta.

Tässä alakohdassa esitetyt ongelmat ja kehityskohteet eivät estä algoritmin käyttöönottoa, sillä ne ovat enemmänkin vain parannuksia nykyiseen ratkaisuun. Niiden avulla algoritmista tulisi entistä monipuolisempi ja sen muokkausmahdollisuudet tapauskohtaisesti paranisivat huomattavasti.

6.2 Ratkaisun skaalautuvuus

Algoritmin ratkaisu skaalautuu hyvin suurempiin määriin tolppia. Viestiliikenteen osalta skaalautuvuutta käsiteltiin kohdassa 4.2 lyhyesti. Viestiliikennettä on mahdollista vielä tehostaa säätämällä sitä, ettei osaa viesteistä lähetetä niin tiheästi. Viestiliikenteen lisäksi myös algoritmi skaalautuu hyvin, sillä Azureen on mahdollista lisätä useita WebJobeja pyörimään rinnakkain. Tällöin vain täytyisi rajata jokaiselle algoritmille tietty viestiliikenne ja tietyt vaiheet, joita kyseinen algoritmi säätelisi. Tällä hetkellä yksi WebJob riittää varsin hyvin pyörittämään kaikkia käytössä olevia tolppia. Ratkaisu kuitenkin skaalautuu hyvin siihen, että tolppia tulisi tuhansia lisää.

6.3 Kuormantasausalgoritmin muut käyttökohteet

Kuormantasausalgoritmin perusideaa, jossa virrankulutusta säädellään sulakkeen palamisen estämiseksi, voidaan viedä muihinkin yhteyksiin kuin vain autojen piiriin. Toinen

käyttökohde voisi olla omakotitalot, joiden sulakkeet eivät kestä esimerkiksi sähkökiukaan ja lattialämmityksen samanaikaista päällä oloa. Tällöin algoritmi voisi sammuttaa lattialämmityksen siksi aikaa kun kiuas on päällä.

Ratkaisun voi myös viedä omakotitalossa asuville sähköautojen lataajille esimerkiksi niin, että sähköauton lataustehoa säädellään sen mukaan, miten muualla omakotitalossa käytetään virtaa. Esimerkiksi kun astianpesukone menee päälle, niin sähköauton lataustehoa lasketaan.

Nämä muut mahdolliset käyttökohteet vaativat algoritmiin pieniä muutoksia ja sen, että vaiheen kokonaisvirrankulutusta voidaan seurata esimerkiksi lisäämällä sähkökaappiin vaiheille virrankulutuksenlukija, joka lähettäisi viestejä algoritmille.

6.4 Katsaus ja vertailu markkinoilla oleviin kuormantasausratkaisuihin

Seuraavaksi verrataan tässä työssä toteutettua kuormantasausalgoritmia ja sen toteutusta muihin markkinoilla oleviin vastaavanlaisiin ratkaisuihin. Samalla pohditaan, miksi tämän työn algoritmin toteutuksessa tehtiin valitunlaisia ratkaisuja. Vertailussa hyödynnetään niitä tietoja, joita tuotteista on kerrottu nettisivuilla tai haastatteluissa. Vertailu ei ole kovin kattava, sillä kaupallisten tuotteiden algoritmit ovat luonnollisesti sellaista tietoa, jota ulkopuoliset eivät pääse tarkastelemaan.

Ensimmäinen verrattava toteutus on Virta-yrityksen tekemä kuormanhallintaratkaisu. Ratkaisu on nimeltään dynaaminen kuormanhallinta ja se jakaa lataustehon tasaisesti lataajien kesken, eli kaikki autot saavat saman verran virtaa. Ratkaisu ei vaadi lisäinvestointeja ja sitä ajetaan pilvipalvelussa. Käyttäjä voi itse asettaa latauspisteryhmälle maksimikuorman. Ratkaisu toimii lähes kaikkien älykkäiden latauslaitteiden kanssa, kunhan laitteet ovat kytkettynä Virran latauspalveluun GPRS-yhteydellä OCPP-protokollalla. [Virta 2018.] Tämä ratkaisu vaatii SIM-kortin latauslaitteeseen. OCPP-protokolla (Open Charge Point Protocol) on yksinkertainen ja ilmainen avoimen standardin ratkaisu, jolla voidaan kommunikoida hallintajärjestelmän ja latauslaitteiden välillä [Open Charge Alliance 2019].

Toinen verrattava toteutus on Maxem, joka on dynaaminen kuormantasaaja. Maxem Home-versio on suunnattu omakotitaloihin. Ratkaisu seuraa verkon virrankulutusta ja säätelee sen perusteella latauslaitteita. Maxem Home-versio sallii kahden latauslaitteen kytkemisen kuormantasaustoteutukseen. Maxem Plus ja Maxem Pro versiot skaalautuvat 100 latauslaitteeseen asti. Maxem asennetaan sähkökaappiin ja se vaatii verkkoyhteyden. [Maxem 2019.] Ratkaisun verkkosivuilta ei selviä sääteleekö sähkökaapissa oleva laite virrankulutusta vai tapahtuuko se esimerkiksi pilvessä. Sivustolla ei myöskään kerrota miten latauslaitteet ja ratkaisu kommunikoivat keskenään.

Kolmas verrattava ratkaisu on PowerFlex-yrityksen Adaptive Load Management eli mukautuva kuormanhallinta-ratkaisu. Yrityksen mielestä muiden kuormantasausratkaisut eivät huomioi tarpeeksi käyttäjiä. Tämän vuoksi PowerFlexin ratkaisu selvittää käyttäjän

tarpeita kysymällä kuinka paljon käyttäjä tarvitsee ajokilometrejä ja mihin aikaan käyttäjä lähtee. Näiden tietojen pohjalta voidaan kuormaa jakaa niin, että käyttäjän tarpeita huomioidaan paremmin, jolloin vältytään tilanteelta, että auto ei olekaan latautunut. Jos esimerkiksi parkkipaikalla on kaksi autoa, joista ensimmäisen tarvitsee lähteä 4 tunnin päästä ja toisen 24 tunnin päästä, niin voidaan säätää kaikki kuorma ensimmäiselle autolle ja tämän jälkeen ladata toinen auto. Ratkaisu käyttää solmuverkkoa (mesh network), jonka avulla latauslaitteet ovat yhteydessä kuormanhallintajärjestelmään. Kuormanhallintajärjestelmä voi esimerkiksi olla parkkihallin sähkökaapissa ja riittää, että yksi latauslaitteista näkee kuormanhallintajärjestelmän. Solmuverkon kautta viestit kulkevat kaikkien latauslaitteiden kautta toisillensa. [Alba 2017.]

Tässä työssä toteutettu kuormantasausalgoritmi rajoittaa käyttäjiänsä progressiivisesti lataustehon mukaan, jolloin eniten lataavia rajoitetaan enemmän ja vähemmän lataavia vähemmän. Monissa ratkaisuissa rajoitukset tehdään rajoittamalla kaikkia latauslaitteita tasaisesti, jolloin kaikki lataavat saman verran. On mahdollista, että ladattavat autot käyttävät eri verran virtaa, koska myös auto säätelee, kuinka paljon virtaa käytetään. Tällöin on mielestäni järkevin ratkaisu laskea rajoitukset käytetyn virran perusteella.

Monissa muissa kuormantasausratkaisuissa on mahdollisuus priorisoida latauslaitteita. Tämän työn toteutuksessa ei ole priorisointia, vaan kaikki tolpat ovat tasavertaisia. Tällainen ratkaisu on mahdollista kuitenkin lisätä osaksi toteutusta, mikäli tulevaisuudessa se koetaan tärkeäksi.

Osa ratkaisuista vaatii, että parkkipaikalle asennetaan laite, joka virrankulutusta säätelee. Tämän työn ratkaisu toimii pilvipalvelusta käsin, jolloin parkkipaikalle ei tarvitse latauslaitteiden ja pistorasioiden lisäksi muuta. Todennäköisesti meidän ratkaisumme voisi viedä muihinkin latauslaitteisiin kuin tässä työssä esitettyihin. Ainoa vaatimus tällä hetkellä on, että latauslaite voi kommunikoida MQTT-tekniikan kautta molempiin suuntiin.

Kaikki markkinoilla olevat ratkaisut, jotka löysin, keskittyivät nimenomaan vain sähköautojen latauslaitteisiin. Tämän työn ratkaisu kattaa myös esilämmitystolpat, mikä vaikuttaa olevan suurin ero muihin ratkaisuihin. Tämä johtuu osittain myös siitä, että kaikkialla maailmassa ei tarvita esilämmitystä.

Tämän työn ratkaisu vaatii pistorasioilta ja latauslaitteilta WiFi-yhteyden, jotta niihin saadaan verkko. Viestintään käytetään MQTT-protokollaa. Tulevaisuudessa on mahdollista vaihtaa näitäkin ratkaisuja toisiin, mikäli koetaan, että parempia ratkaisuja tähän tarkoitukseen löytyy. Tällä hetkellä näitä käytetään algoritmissa, koska aikaisemmin toteutetut osat käyttivät jo niitä, joten ne oli valmiiksi toteutettu.

Vaikka pieniä eroja eri ratkaisujen välillä on, pääidea on kuitenkin kaikissa sama. Suurimmat eroavaisuudet ovat pääasiassa siinä, miten tolppien ja algoritmien välinen yhteys hoidetaan ja onko algoritmi pilvessä vai fyysinen laite parkkipaikalla.

7 Yhteenveto

Tässä tutkielmassa toteutettiin kuormantasausalgoritmi, jolla voidaan säädellä sähköautojen latauslaitteita sekä esilämmitystolppia, jotta verkko ei ylikuormitu. Esittelen tutkielmassani kuormantasausongelmaa esimerkkiskenaarioiden avulla, algoritmin toteutusta vuokaavioiden avulla, testausta vaatimusmäärittelyn kautta ja lisäksi analysoin ongelmatilanteita ja kehityskohteita algoritmiin liittyen. Suurin ero muihin markkinoilla oleviin kuormantasausratkaisuihin on siinä, että tämän tutkielman algoritmi kattaa myös esilämmitystolpat. Näitä tolppia voidaan kuitenkin käyttää myös sähköautojen hitaaseen lataamiseen.

Työn algoritmissa on vielä kehityskohteita, joita tulee ratkaista, ennen kuin se voidaan ottaa kaupalliseen käyttöön. Ensimmäinen versio kuitenkin osaa lähes kokonaan toimia odotusten mukaisesti, kuten luvun 5 testeissä huomattiin. Algoritmi osaa siis laskea ja nostaa vaiheen kokonaisvirrankulutusta silloin kun sellaisia toimenpiteitä tarvitaan. Tämän lisäksi algoritmi osaa tehdä toimenpiteet oikeassa järjestyksessä eli laskea virrankulutusta ensin rajoittamalla ja sammuttamalla EVSE-tolppia ja sen jälkeen sammuttamalla POW- tai HEAT-tolppia sekä nostaa virrankulutusta käynnistämällä ja poistamalla rajoituksia EVSE-tolpilta.

Sähköautot ovat hyvin todennäköisesti suuri osa liikenteen tulevaisuutta. Niiden myötä myös sähköverkoissa tulee tapahtumaan muutoksia. Sähköverkoissa voidaan hyödyntää erilaisia älyratkaisuja, kuten esimerkiksi ottamalla sähköautojen akuista virtaa takaisin verkkoon virrankulutuspiikkien aikana. Latausjärjestelmät ja -paikat ovat suuressa roolissa, mikäli asetetut sähköajoneuvotavoitteet halutaan saavuttaa. Ihmiset eivät voi hankkia sähköajoneuvoja, jos niitä ei ole mahdollista ladata. Tästä syystä taloyhtiöiden tulee tehdä ratkaisuja, jotta myös kerrostaloissa asuvat saavat sähköajoneuvonsa ladattua. Samaan aikaan moni omakotitalossa asuva sähköajoneuvon omistaja kärsii sähköverkon ylikuormittumisesta. Molempiin ongelmiin ratkaisuksi voidaan tarjota kuormantasausalgoritmia, joka on sähköjärjestelmän uudelleen rakentamiseen verrattuna edullinen ratkaisu. Kuormantasauksen merkitys kasvaa sitä mukaa, kun sähköautojen määrä maailmalla kasvaa.

8 Viiteluettelo

Michael Alba. 2017. *PowerFlex says its Adaptive Load Management is the best technology to balance EV charging loads*. https://chargedevs.com/features/powerflex-says-its-adaptive-load-management-is-the-best-technology-to-balance-ev-charging. Charged EVs. Viitattu 13.3.2019.

Autoalan Tiedotuskeskus. 2018. *Ennusteet*. http://www.aut.fi/tilastot/ennusteet. Viitattu 13.3.2019.

Autoalan Tiedotuskeskus. 2019. *Liikennekäytössä olevan autokannan kehitys*. http://www.aut.fi/tilastot/autokannan_kehitys/ajoneuvokannan_kehitys. Viitattu 13.3.2019.

Thomas Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein. 2009. *Introduction to Algorithms*. MIT Press.

Onur Elma, Ugur Savas Selamogullari. 2015. *A new home energy management algorithm with voltage control in a smart home environment*. Energy 91, 720-731.

Paddy Finn, Colin Fitzpatrick, David Conolly. 2012. *Demand side management of electric car charging: Benefits for consumer and grid*. Energy 42, 1, 358-363.

Gaston C. Hillar. 2017. *MQTT Essentials – A Lightweight IoT Protocol*. Packt Publishing.

IEA. 2018. *Global EV Outlook 2018.*

Karol Lina Lopez, Christian Gagne, Marc-Andre Gardner. 2018. *Demand-Side Management using Deep Learning for Smart Charging of Electric Vehicles*. IEEE Transactions on Smart Grid 10, 3, 2683-2691.

Maxem. 2019. *Dynamic Load Balancing Prevent Grid Overload. https://maxem.io/en/dynamic-load-balancing-prevent-grid-overload*. Viitattu 19.4.2019.

Peter Membrey, David Hows, Eelco Plugge. 2012. *Practical Load Balancing*. Springer Verlag.

Petra Mesaric, Slavko Krajcar. 2015. *Home demand side management integrated with electric vehicles and renewable energy sources*. Energy and Buildings, 108, 1-9.

Microsoft. 2018a. *Run Background tasks with WebJobs in Azure App Service.* https://docs.microsoft.com/en-us/azure/app-service/webjobs-create. Viitattu 21.3.2019.

Microsoft. 2018b. *Azure Functions C# developer reference*. https://docs.microsoft.com/en-us/azure/azure-functions/functions-dotnet-class-library. Viitattu. 18.4.2019.

Microsoft. 2019. *Dictionary<TKey,TValue> Class*. https://docs.microsoft.com/en-us/dotnet/api/system.collections.generic.dictionary-2?view=netframework-4.7.2. Viitattu 12.4.2019.

Motiva. 2015. *Kiinteistöjen latauspaikat -esiselvitys*. https://www.motiva.fi/files/10869/Kiinteistojen_latauspaikat_-esiselvitys.pdf. Viitattu 8.3.2019.

Open Charge Alliance. 2019. *Open Charge Alliance.* https://www.openchargealliance.org/. Viitattu 19.4.2019.

Plugit. 2019. *Latauspistoketyypit sähköautoille*. https://plugit.fi/fi-fi/article/etusivu/latauspistoketyypit-sahkoautoille/135/. Viitattu 12.3.2019.

RabbitMQ. 2019. *RabbitMQ.* https://www.rabbitmq.com/. Viitattu 22.3.2019.

Sesko. 2018. *Sähköajoneuvojen lataussuositus*. https://www.sesko.fi/files/889/Lataussuositus_2018_2018-03-08.pdf. Viitattu 8.3.2019.

Steven Skiena. 2008. *The Algorithm Design Manual*. Springer.

Teknologiateollisuus. 2019. *Sähköisen liikenteen tilannekatsaus Q4/2018***.** https://emobility.teknologiateollisuus.fi/sites/emobility/files/file_attachments/sahkoinen_liikenne_tilannekatsaus_2018_q4_20190214_jaettava.pdf. Viitattu 12.3.2019.

Valtioneuvosto. 2017. *Liikenteen vaihtoehtoisten käyttövoimien jakeluverkko. Suomen kansallinen ohjelma.* http://urn.fi/URN:ISBN:978-952-243-501-9. Viitattu 12.3.2019.

Virta. 2018. *Lataa turvallisesti: Dynaaminen kuormanhallinta huolehtii kiinteistön sähkökapasiteetista.* https://www.virta.global/news-fi/dynaaminen-kuormanhallinta-ja-s%C3%A4hk%C3%B6auton-lataus. Viitattu 19.4.2019.

Geoff Webber-Cross. 2014. *Learning Microsoft Azure*. Packt Publishing.

Letterio Zuccaro, Alessandro Di Giorgio, Francesco Liberati, Silvia Canale, Andrea Lanna, Victor Fernandez Pallares, Alejandro Martinez Blanco, Raúl Urbano Escobar, Jure Ratej, Borut Mehle, Ursula Krisper. 2014. *Smart Vehicle to Grid Interface Project: Electromobility Management System Architecture and Field Test Results.* 2014 IEEE International Electric Vehicle Conference, 1-7.

Liite 1: Testien tulosteet

Testi 1.1

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 16 / Kokonaisvirrankulutus: 16 / Virrankulutuspyrkimys: 12
1 EVSE / Maksimivirta: 16 / Virta: 16 / Rajoitus: 0

Lopputilanne:
Sulake: 16 / Kokonaisvirrankulutus: 12 / Vähennys: 4
1 EVSE / Maksimivirta: 16 / Virta: 12 / Rajoitus: 4

Suoritusaika: 29,7546 ms

Testi 1.2

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 16 / Kokonaisvirrankulutus: 16 / Virrankulutuspyrkimys: 12
1 EVSE / Maksimivirta: 10 / Virta: 10 / Rajoitus: 0
2 EVSE / Maksimivirta: 10 / Virta: 6 / Rajoitus: 4

Lopputilanne:
Sulake: 16 / Kokonaisvirrankulutus: 12 / Vähennys: 4
1 EVSE / Maksimivirta: 10 / Virta: 6 / Rajoitus: 4
2 EVSE / Maksimivirta: 10 / Virta: 6 / Rajoitus: 4

Suoritusaika: 33,3275 ms

Testi 1.3

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 32 / Kokonaisvirrankulutus: 34 / Virrankulutuspyrkimys: 26
1 EVSE / Maksimivirta: 16 / Virta: 16 / Rajoitus: 0
2 EVSE / Maksimivirta: 16 / Virta: 10 / Rajoitus: 6
3 EVSE / Maksimivirta: 16 / Virta: 8 / Rajoitus: 8

Lopputilanne:
Sulake: 32 / Kokonaisvirrankulutus: 26 / Vähennys: 8
1 EVSE / Maksimivirta: 16 / Virta: 11 / Rajoitus: 5
2 EVSE / Maksimivirta: 16 / Virta: 8 / Rajoitus: 8
3 EVSE / Maksimivirta: 16 / Virta: 7 / Rajoitus: 9

Suoritusaika: 28,7095 ms

Testi 2.1

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 16 / Kokonaisvirrankulutus: 18 / Virrankulutuspyrkimys: 14
1 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / Käynnistysaika: 16.4.2019 15.38.24 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / Käynnistysaika: 15.4.2019 15.38.24 / OnkoSammutettu: False
3 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / Käynnistysaika: 14.4.2019 15.38.24 / OnkoSammutettu: False

Lopputilanne:
Sulake: 16 / Kokonaisvirrankulutus: 12 / Vähennys: 6
1 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / Käynnistysaika: 16.4.2019 15.38.24 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / Käynnistysaika: 15.4.2019 15.38.24 / OnkoSammutettu: False
3 EVSE / Maksimivirta: 16 / Virta: 0 / Rajoitus: 10 / Käynnistysaika: 14.4.2019 15.38.24 / OnkoSammutettu: True

Suoritusaika: 40,2459 ms

Testi 2.2

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 32 / Kokonaisvirrankulutus: 31 / Virrankulutuspyrkimys: 8
1 EVSE / Maksimivirta: 16 / Virta: 16 / Rajoitus: 0 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 15 / Virta: 15 / Rajoitus: 1 / OnkoSammutettu: False

Lopputilanne:
Sulake: 32 / Kokonaisvirrankulutus: 0 / Vähennys: 31
1 EVSE / Maksimivirta: 16 / Virta: 0 / Rajoitus: 0 / OnkoSammutettu: True
2 EVSE / Maksimivirta: 15 / Virta: 0 / Rajoitus: 1 / OnkoSammutettu: True

Suoritusaika: 19,1035 ms

Testi 3.1

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 16 / Kokonaisvirrankulutus: 15 / Virrankulutuspyrkimys: 13
1 POW / Virta: 5 / Käynnistysaika: 16.4.2019 16.15.57
2 HEAT / Virta: 5 / Käynnistysaika: 15.4.2019 16.15.57
3 HEAT / Virta: 5 / Käynnistysaika: 14.4.2019 16.15.57

Lopputilanne:
Sulake: 16 / Kokonaisvirrankulutus: 10 / Vähennys: 5
1 POW / Virta: 5 / Käynnistysaika: 16.4.2019 16.15.57
2 HEAT / Virta: 5 / Käynnistysaika: 15.4.2019 16.15.57
3 HEAT / Virta: 0 / Käynnistysaika: 14.4.2019 16.15.57

Suoritusaika: 20,0129 ms

Testi 3.2

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 16 / Kokonaisvirrankulutus: 16 / Virrankulutuspyrkimys: 4
1 HEAT / Virta: 4 / Käynnistysaika: 16.4.2019 16.32.10
2 HEAT / Virta: 4 / Käynnistysaika: 15.4.2019 16.32.10
3 HEAT / Virta: 4 / Käynnistysaika: 14.4.2019 16.32.10
4 POW / Virta: 4 / Käynnistysaika: 13.4.2019 16.32.10

Lopputilanne:
1 HEAT / Virta: 4 / Käynnistysaika: 16.4.2019 16.32.10
2 HEAT / Virta: 0 / Käynnistysaika: 15.4.2019 16.32.10
3 HEAT / Virta: 0 / Käynnistysaika: 14.4.2019 16.32.10
4 POW / Virta: 0 / Käynnistysaika: 13.4.2019 16.32.10
Sulake: 16 / Kokonaisvirrankulutus: 4 / Vähennys: 12

Suoritusaika: 27,4672 ms

Testi 4.1:

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 16 / Kokonaisvirrankulutus: 16 / Virrankulutuspyrkimys: 6
1 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / Käynnistysaika: 16.4.2019 16.56.52 / OnkoSammutettu: False
2 HEAT / Virta: 5 / Käynnistysaika: 15.4.2019 16.56.52
3 HEAT / Virta: 5 / Käynnistysaika: 14.4.2019 16.56.52

Lopputilanne:
Sulake: 16 / Kokonaisvirrankulutus: 5 / Vähennys: 11
1 EVSE / Maksimivirta: 16 / Virta: 0 / Rajoitus: 16 / Käynnistysaika: 16.4.2019 16.56.52 / OnkoSammutettu: True
2 HEAT / Virta: 5 / Käynnistysaika: 15.4.2019 16.56.52
3 HEAT / Virta: 0 / Käynnistysaika: 14.4.2019 16.56.52

Suoritusaika: 32,5689 ms

Testi 4.2

Toimenpide: laske virrankulutusta

Alkutilanne:
Sulake: 32 / Kokonaisvirrankulutus: 32 / Virrankulutuspyrkimys: 9
1 EVSE / Maksimivirta: 16 / Virta: 9 / Rajoitus: 8 / Käynnistysaika: 16.4.2019 17.14.41 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 16 / Virta: 8 / Rajoitus: 8 / Käynnistysaika: 15.4.2019 17.14.41 / OnkoSammutettu: False
3 POW / Virta: 5 / Käynnistysaika: 14.4.2019 17.14.41
4 HEAT / Virta: 5 / Käynnistysaika: 15.4.2019 17.14.41
5 HEAT / Virta: 5 / Käynnistysaika: 13.4.2019 17.14.41

Lopputilanne:
Sulake: 32 / Kokonaisvirrankulutus: 5 / Vähennys: 27
1 EVSE / Maksimivirta: 16 / Virta: 0 / Rajoitus: 8 / Käynnistysaika: 16.4.2019 17.27.39 / OnkoSammutettu: True
2 EVSE / Maksimivirta: 16 / Virta: 0 / Rajoitus: 8 / Käynnistysaika: 15.4.2019 17.27.39 / OnkoSammutettu: True
3 POW / Virta: 0 / Käynnistysaika: 14.4.2019 17.27.39
4 HEAT / Virta: 5 / Käynnistysaika: 15.4.2019 17.27.39
5 HEAT / Virta: 0 / Käynnistysaika: 13.4.2019 17.27.39

Suoritusaika: 30,3574 ms

Testi 5.1:

Toimenpide: nosta virrankulutusta.

Alkutilanne:
Sulake: 32 / Kokonaisvirrankulutus: 6
1 EVSE / Maksimivirta: 16 / Virta: 0 / Rajoitus: 6 / OnkoSammutettu: True
2 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / OnkoSammutettu: False

Lopputilanne:
Sulake: 32 / Kokonaisvirrankulutus: 12
1 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / OnkoSammutettu: False

Suoritusaika: 13,8612 ms

Testi 5.2:

Toimenpide: nosta virrankulutusta

Alkutilanne:
Sulake: 16 / Kokonaisvirrankulutus: 8
1 EVSE / Maksimivirta: 16 / Virta: 0 / Rajoitus: 6 / OnkoSammutettu: True
2 EVSE / Maksimivirta: 16 / Virta: 8 / Rajoitus: 8 / OnkoSammutettu: False

Lopputilanne:
Sulake: 16 / Kokonaisvirrankulutus: 8
1 EVSE / Maksimivirta: 16 / Virta: 0 / Rajoitus: 6 / OnkoSammutettu: True
2 EVSE / Maksimivirta: 16 / Virta: 8 / Rajoitus: 8 / OnkoSammutettu: False

Suoritusaika: 38,3672 ms

Testi 6.1:

Toimenpide: nosta virrankulutusta.

Alkutilanne:
Sulake: 32 / Kokonaisvirrankulutus: 12
1 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / OnkoSammutettu: False

Lopputilanne:
Sulake: 32 / Kokonaisvirrankulutus: 16
1 EVSE / Maksimivirta: 16 / Virta: 8 / Rajoitus: 8 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 16 / Virta: 8 / Rajoitus: 8 / OnkoSammutettu: False

Suoritusaika: 22,7679 ms

Testi 6.2:

Toimenpide: nosta virrankulutusta

Alkutilanne:
Sulake: 40 / Kokonaisvirrankulutus: 24
1 EVSE / Maksimivirta: 16 / Virta: 10 / Rajoitus: 6 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 16 / Virta: 8 / Rajoitus: 8 / OnkoSammutettu: False
3 EVSE / Maksimivirta: 16 / Virta: 6 / Rajoitus: 10 / OnkoSammutettu: False

Lopputilanne:
Sulake: 40 / Kokonaisvirrankulutus: 31
1 EVSE / Maksimivirta: 16 / Virta: 12 / Rajoitus: 4 / OnkoSammutettu: False
2 EVSE / Maksimivirta: 16 / Virta: 10 / Rajoitus: 6 / OnkoSammutettu: False
3 EVSE / Maksimivirta: 16 / Virta: 9 / Rajoitus: 7 / OnkoSammutettu: False

Suoritusaika: 54,0348 ms